Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 917042, 9 pages
http://dx.doi.org/10.1155/2015/917042

Research Article

Time Synchronization for Wireless Sensor Networks Using Adaptive Linear Prediction

Yulong Xing, Yongrui Chen, Weidong Yi, and Chenghua Duan

University of Chinese Academy of Sciences, Beijing 100049, China

Correspondence should be addressed to Yongrui Chen; chenyr@ucas.ac.cn

Received 29 September 2014; Accepted 26 January 2015

Academic Editor: Chase Wu

Time synchronization is a crucial component in wireless sensor networks (WSN), especially for location-aware applications. The precision of time-based localization algorithms is closely related to the accuracy of synchronization. The estimation of synchronization errors in most of the existing time synchronization algorithms is based on some statistical distribution models. However, these models may not be able to accurately describe the synchronization errors due to the uncertainties in clock drift and message delivery delay in synchronization. Considering that the synchronization errors are highly temporally correlated (short-term correlation), in this paper, we present an adaptive linear prediction synchronization (ALPS) scheme for WSN. By applying linear prediction on synchronization errors and adaptively adjusting prediction coefficients based on the difference between the estimated values and the real values, ALPS enhances synchronization accuracy at a relatively low cost. ALPS has been implemented on the Tmote-sky platform. As experiment results demonstrate, compared with RBS and TPSN, ALPS cuts synchronization cost by almost 50% while achieving the same accuracy; compared with DMTS and PulseSync, ALPS reduces the MSE (mean square error) of synchronization errors by 41% and 24%, respectively, with the same cost.

1. Introduction

Time synchronization is one of the key middle-ware components in wireless sensor networks (WSN) [1]. Accurate time synchronization is critical to many tasks in various WSN applications including duty cycling scheduling, data fusion and tracking, and transmission scheduling. Localization is another important application of time synchronization. The accuracy of time synchronization directly influences the precision of localization [2].

Among the existing localization algorithms [3–6], the most accurate localization approaches are ranged-based methods. These methods usually use the time-of-arrival (TOA) [7], time-difference-of-arrival (TDOA) [8, 9], angle-of-arrival (AOA) [10], and received signal strength (RSS) [6] to compute the distances between nodes and anchors and then obtain the location information. Among the above four methods, TOA and TDOA are time-based localization algorithms, which are promising ranging methods due to their high accuracy and low cost. The simulation results [2] demonstrate that the accuracy of the time synchronization is a central issue in time-based localization approaches. Therefore, time synchronization plays a critical role in localization algorithms.

Many time synchronization algorithms have been proposed in recent years. The early time synchronization algorithms, such as RBS [11], TPSN [12], DMTS [13], and FTSP [14], are all based on timestamp exchanges, which are intended to offset the uncertainties of message delivery delay to enhance the synchronization accuracy. On-demand synchronization (ODS) [15] adaptively adjusts the synchronization interval for customized accuracy with minimized communication overhead. Lenzen et al. propose PulseSync [16] which distributes information on clock values as fast as possible and achieves a higher synchronization accuracy than FTSP. Leng and Wu present a synchronization scheme under unknown Gaussian distribution aiming at achieving both low computational complexity and high performance [17]. Leng and Wu also model the WSN synchronization under exponential delays as a linear programming problem and solve it with a novel low-complexity estimator [18]. In [19], a Kalman filter is used to estimate both clock offset and skew. An improvement on [19] by adopting an adaptive multimodel mechanism is presented in [20]. Zheng and Wu present a unified framework to jointly solve time synchronization and localization problems at the same time [2]. Jin et al. study how voltage influences the clock skew and proposes a novel voltage-aware time synchronization (VATS) scheme [21].

Most of existing synchronization schemes assume that the distribution of clock skew and the delivery delay of synchronization packets follow some statistical models (such as Gaussian distribution model in [15, 17, 20, 21], exponential distribution model in [18, 22, 23], and gamma distribution model in [1]). However, the clock drift and message delivery delay are essentially very random and easily affected by some environment factors (such as temperature and voltage). Therefore, it is not easy to describe them accurately through any predetermined model [1, 24]. Moreover, some algorithms need to calibrate the clock skew frequently (such as [11, 14, 20, 22]). However, as [15] demonstrates, frequent clock skew calibration will not only increase the computation and communication overhead, but also may increase the synchronization errors due to the miscalculation of the message delivery delay.

Linear prediction is a mathematical operation where future values of a discrete-time signal are estimated as a linear function of previous samples. In digital signal processing, linear prediction is often called linear predictive coding (LPC) and is widely used in speech coding, digital filter, and wireless channel estimation. Considering that synchronization errors are highly temporally correlated (as Figure 3 shows), in this paper, an adaptive linear predication synchronization (ALPS) scheme for WSN is proposed. To the best of our knowledge, this is the first work which applies linear prediction to the time synchronization in WSN.

ALPS predicts clock offset based on its historical values. ALPS does not assume the clock offset to follow any particular statistical distribution or need frequent clock skew calibration. As experimental results demonstrate, ALPS has a better performance than DMTS and PulseSync without increasing synchronization cost; ALPS reduces synchronization cost compared to RBS and TPSN while maintaining the same synchronization accuracy.

2. Motivation

In WSN, each node has an independent local clock. Ideally, the clock should satisfy the equation $C(t) = t$, where t stands for the ideal or reference time. However, due to the influence of voltage, temperature, and other environmental factors, the clock will drift away from the ideal time. The clock model is illustrated in Figure 1. In general, the clock function of node i can be modelled as

$$C_i(t) = \theta + f \cdot t, \tag{1}$$

where θ and f are clock offset and clock skew, respectively.

In WSN, offsets exist between clocks of different nodes due to the difference in clock skew and the initial phase.

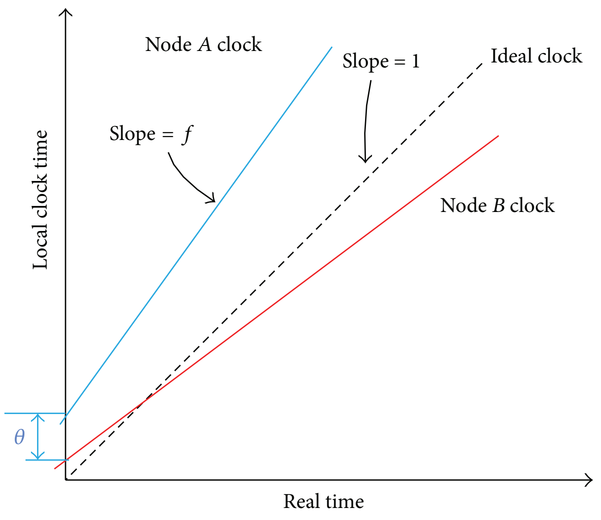

FIGURE 1: Clock model of sensor nodes.

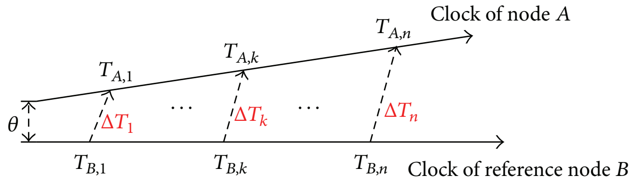

FIGURE 2: Synchronization model.

Consider the time synchronization process between node A and node B, where B is the reference node and A is the node to be synchronized, as illustrated in Figure 2. Suppose node B sends a message to node A about its clock reading $T_{B,k}$ at node B's local time $T_{B,k}$ and node A receives node B's message at A's local time $T_{A,k}$. Then $T_{A,k}$ and $T_{B,k}$ satisfy the following expression:

$$T_{A,k} = f \cdot T_{B,k} + \theta + \tau, \tag{2}$$

where θ and f are the relative clock offset and skew between node A and node B, respectively. τ is the delay of message transmitted from node B to node A.

The goal of time synchronization is to make the clocks of node A and node B the same. In (2), there are three parameters, f, θ, and τ to be estimated. For f and τ, most of the synchronization schemes first assume that they follow a particular statistical distribution and then estimate them by some statistical means. For example, the least squares (LS) estimator is applied in [14, 24]. In [18, 23], the parameters are estimated using the maximum likelihood estimator (MLE) and minimum variance unbiased estimator (MVUE), respectively.

However, the delay of the message for synchronization (τ) consists of many components, including send/receive time, access time, transmission/reception time, interrupt time, and encoding/decoding time. Among them, the send/receive time, access time, and interrupt time are nondeterministic. Although the nondeterminacy of the synchronization delay can be reduced by inserting the timestamp into the appropriate place in the packet at the lower layers [13, 14], it can not be eliminated completely. On the other hand, clock skew (f), affected by voltage, temperature, and other environment factors [21], exhibits great randomness and is difficult to be described by any particular statistical model. Moreover, since the randomness of clock skew and synchronization delay are not independent of each other, inappropriate estimation not only can not eliminate the errors, but also introduce new errors [15]. Therefore, there are limitations when using a specific statistical distribution model in time synchronization. Linear prediction, however, as a widely used method in the field of random signal estimation, is independent of any special statistical distribution model. Considering that the clock skew changes slowly and the synchronization error samples exhibit high short-term temporal correlation and periodicity, we adopt adaptive linear prediction as the estimator of the synchronization errors in our synchronization scheme.

3. Algorithm

3.1. Linear Prediction Model. The key idea of linear prediction is that if there is temporal correlation between signal samples, we can use past sample values to predict current and future sample values. The sample to be predicted can be closely approximated as a linear combination of past samples. The prediction coefficients are determined by minimizing a certain function of differences between sample values and predicted values.

In the time synchronization model, if we consider the clock offset caused by both clock skew and the synchronization delay as a single variable ΔT_k, $T_{A,k}$, and $T_{B,k}$ satisfy the following expression:

$$T_{A,k} = T_{B,k} + \Delta T_k, \tag{3}$$

where ΔT_k is the sum of clock offset caused by clock skew and transmission delay at the Kth synchronization.

In order to examine how the synchronization errors change with time, we implemented several experiments (lasting 24 hours, with synchronization interval 2 s, 3 groups) on the Tmote-sky platform. In these experiments, we compensated the clock offset with a fixed value ΔT_k and recorded all synchronization errors during the testing time. The synchronization errors over time are shown in Figure 3, which is similar to [15]. As Figure 3 demonstrates, in short term (within 30–50 samples), synchronization errors do not vary much (short-term correlation), due to the relatively stable clock skew, where small fluctuations result from rounding in calculation. In long term (interval more than 100–120 samples), synchronization errors become periodical (long-term correlation), due to the effect of varied clock skew. These experiment results demonstrate that the synchronization errors are highly temporally correlated. Therefore, linear prediction, as a simple and widely used prediction method, can be utilized to estimate ΔT_k. Also, the prediction coefficients can be adaptively assigned to minimize the MSE of synchronization errors.

The basic idea of linear prediction is that ΔT_k can be approximated by its historical values $(\Delta T_{k-1}, \Delta T_{k-2}, \ldots)$.

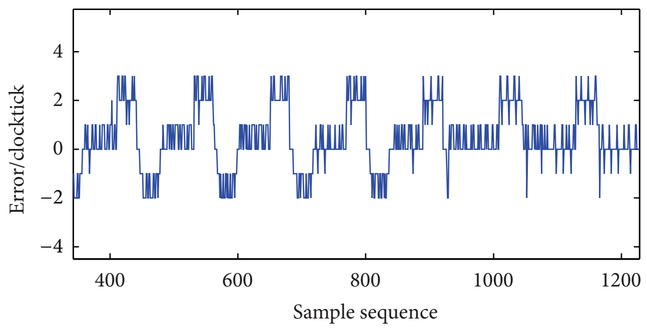

FIGURE 3: Synchronization errors over time using a fixed compensation value.

We define two symbols: (i) $\widehat{\Delta T_k}$, which is the predicted value of ΔT_k and (ii) e_k, which is the prediction error, that is, the difference between the real value ΔT_k and the predicted value $\widehat{\Delta T_k}$. ΔT_k is the real time difference between the master node (node B) and the slave node (node A), where the former value is read in the synchronization packet, and the latter is the local time of the slave node when the synchronization packet arrives. Then, $\widehat{\Delta T_k}$ and e_k can be expressed as

$$\widehat{\Delta T_k} = a_1 \Delta T_{k-1} + \cdots + a_p \Delta T_{k-p} = \sum_{i=1}^{p} a_i \Delta T_{k-i}, \tag{4}$$

$$e_k = \Delta T_k - \widehat{\Delta T_k} = \Delta T_k - \sum_{i=1}^{p} a_i \Delta T_{k-i}, \tag{5}$$

where a_i $(i = 1, 2, \ldots, p)$ are the prediction coefficients and p is the prediction order.

3.2. Calculation of the Prediction Coefficient a_i. We use the synchronization mean square error (MSE) to evaluate the synchronization performance. Smaller MSE indicates higher synchronization accuracy [22]. The E_n, which is based on MSE, is defined as follows:

$$\begin{aligned} E_n &= \sum_n e_k^2 = \sum_n \left(\Delta T_k - \widehat{\Delta T_k}\right)^2 \\ &= \sum_n \left(\Delta T_k - \sum_{i=1}^{p} a_i \Delta T_{k-i}\right)^2. \end{aligned} \tag{6}$$

To minimize E_n, set the partial derivative of E_n to a_i $(i = 1, 2, \ldots, p)$ be zero; that is, $\partial E_n / \partial a_j = 0$. Consider

$$2\sum_n \Delta T_k \Delta T_{k-j} - 2\sum_{i=1}^{p} a_i \sum_n \Delta T_{k-i} \Delta T_{k-j} = 0, \tag{7}$$

where $j = 1, 2, \ldots, p$.

Equation (7) can be further reduced to the following equation:

$$\sum_n \Delta T_k \Delta T_{k-j} = \sum_{i=1}^{p} a_i \sum_n \Delta T_{k-i} \Delta T_{k-j}. \tag{8}$$

```
Calculate $\widehat{\Delta T}_k$
**Inputs:** $\Delta T_{k-1}, \Delta T_{k-2}, \ldots, \Delta T_{k-p}$
**Output:** $\widehat{\Delta T}_k$
(1) $\Phi(i,j) = f_1(\Delta T_{k-1}, \Delta T_{k-2}, \ldots, \Delta T_{k-p})$
(2) $R(j) = f_2(\Delta T_{k-1}, \Delta T_{k-2}, \ldots, \Delta T_{k-p})$
(3) $Y - W(P) = f_3(R(j), \Phi(i,j))$
(4) $a_1, a_2, \ldots, a_p = f_4(L - D(p))$
(5) $\widehat{\Delta T}_k = f_5(\Delta T_{k-1}, \Delta T_{k-2}, \ldots, \Delta T_{k-p}, a_1, a_2, \ldots, a_p)$
```

Algorithm 1: The way to calculate $\widehat{\Delta T}_k$.

Apparently, by solving this equation set, we can obtain all the prediction coefficients a_i $(i = 1, 2, \ldots, p)$ which minimize E_n.

Define $\Phi(i,j)$ and $R(j)$ as follows:

$$\Phi(i,j) = \sum_n \Delta T_{k-i} \Delta T_{k-j}, \quad (i, j = 1, 2, \ldots, p),$$
$$R(j) = \sum_n \Delta T_k \Delta T_{k-j}, \quad (j = 1, 2, \ldots, p), \tag{9}$$

where $R(j)$ is the autocorrelation function of ΔT_k.

Combining (8) and (9) and the properties of the autocorrelation function, we will have

$$\sum_{i=1}^{p} a_i R(|i-j|) = R(j), \quad (j = 1, 2, \ldots, p) \tag{10}$$

expressed in matrix form as follows:

$$\begin{bmatrix} R(0) & R(1) & \cdots & R(p-1) \\ R(1) & R(0) & \cdots & R(p-2) \\ \vdots & \vdots & \vdots & \vdots \\ R(p-1) & R(p-2) & \cdots & R(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} R(1) \\ R(2) \\ \vdots \\ R(p) \end{bmatrix}. \tag{11}$$

The above is called Yule-Walker equations, where the coefficient matrix is a $p \times p$ Toeplitz matrix in which each descending diagonal from left to right is constant.

We can use Levinson-Durbin recursion algorithm to solve Yule-Walker equations. The Levinson-Durbin recursion algorithm is given as follows [25].

(1) $E_0 = R(0)$, when $i = 0$.

(2) Perform recursive calculation with the following formulas:

$$k_i = \frac{R(i) - \sum_{j=1}^{i-1} a_j^{i-1} R(i-j)}{E_{i-1}}, \quad 1 \le i \le p,$$
$$a_i^{(i)} = k_i,$$
$$a_j^{(i)} = a_j^{i-1} - k_i a_{i-j}^{(i-1)}, \quad 1 \le j \le i-1,$$
$$E_i = (1 - k_i^2) E_{i-1}. \tag{12}$$

(3) $i = i + 1$. If $i > p$, then go to (4), or return to (2).

(4) The final solution is given as follows:

$$a_j = a_j^{(p)}, \quad 1 \le j \le p. \tag{13}$$

3.3. Algorithm Description. ALPS is given as follows. First, a routing tree is established in the network initialization stage, where SINK node acts as its root. SINK node periodically broadcasts synchronization packets, which include timestamp, to the lower levels of the tree. When a child node receives the packet, it extracts the timestamp and calculates the difference (clock offset, which is used as the input of ALPS) between the local clock and the timestamp and rebroadcasts the synchronization packet only with its own local timestamp. Then, the child node calculates the prediction coefficients a_i according to previous clock offset samples $\Delta T_{k-1}, \Delta T_{k-2}, \ldots, \Delta T_{k-p}$, as Section 3.2 describes. ALPS distributively runs on every node independently. Each node calculates its prediction coefficients based on the measurements of its own clock errors and calibrates its clock independently. For the first p samples, when estimating ΔT_k $(k = p + 1)$, we just set $a_i = 1$ for $i = 1$ and $a_i = 0$ for $i = 2, \ldots, k-1$. Finally, we substitute a_i $(i = 1, 2, \ldots, p)$ and $\Delta T_{k-1}, \Delta T_{k-2}, \ldots, \Delta T_{k-p}$ into (4) to predict ΔT_k. $\widehat{\Delta T}_k$ is used to calibrate the child node's local clock, in order to make the clocks between the parent node and the child node relatively synchronized right before the next synchronization cycle.

The method to calculate $\widehat{\Delta T}_k$ at Kth synchronization is given in Algorithm 1.

ALPS is described as follows.

(1) The child node maintains a FIFO queue which consists of p registers. The previous p clock offset samples are put into the registers by the order of $\Delta T_{k-p}, \ldots, \Delta T_{k-2}, \Delta T_{k-1}$.

(2) With (9), $\Phi(i,j)$ and the autocorrelation function $R(j)$ of clock offsets can be calculated.

(3) With $\Phi(i,j)$ and $R(j)$, Yule-Walker equations can be created.

(4) Using Levinson-Durbin iterative algorithm, calculate the prediction coefficients $a_1, a_2, \ldots, a_p$.

(5) Calculate the next clock offset $\widehat{\Delta T}_k$ by (4).

FIGURE 4: Indoor testbed.

(6) Use $\widehat{\Delta T_k}$ for local clock calibration.

(7) When the next synchronization packet arrives, calculate ΔT_k by (3) according to the node's local time and the timestamp which is extracted from the synchronization packet.

(8) Insert ΔT_k at the end of FIFO queue, and go back to step (2).

The complexity of ALPS algorithm is analyzed as follows. The most complex part of ALPS is the Levinson-Durbin algorithm. Within the Levinson-Durbin algorithm, the most complicated calculation is resolving the k_i, which requires a nested loop with complexity $O(p^2)$, where p is the prediction order. Therefore, the complexity of the whole ALPS is $O(p^2)$.

4. Performance Evaluation

To evaluate the performance of ALPS, we implemented it in a real testbed, as shown in Figure 4. In our experiments, we adopt CTP (collection tree protocol, commonly used in WSN), as the routing algorithm on the nodes. Each node looks for a parent node to be synchronized and transfers data packets to the SINK node hop by hop through this data collection tree, thus forming a tree-based topology. The target hardware platform is Tmote-sky node, and the software platform is Contiki, an open source lightweight embedded operating system widely used in WSN. The Tmote-sky node features an MSP430 MCU and a CC2420 radio transceiver, and the clock frequency is 32768 Hz. In the following experiments, we synchronize the local time generated by the 32768 Hz clock. Thus, one clock tick is equal to 1/32768 s. The Tmote-sky supports complicated calculations and has the ability to store plenty of external data. These advantages of Tmote-sky facilitate the ALPS implementation.

In order to obtain the clock readings of parent nodes and their children nodes at the same time, a pulse signal was sent periodically to these nodes. The nodes recorded and output their clock readings upon receiving the pulse signal. In this way, we can periodically sample the clocks of the synchronization nodes to be tested.

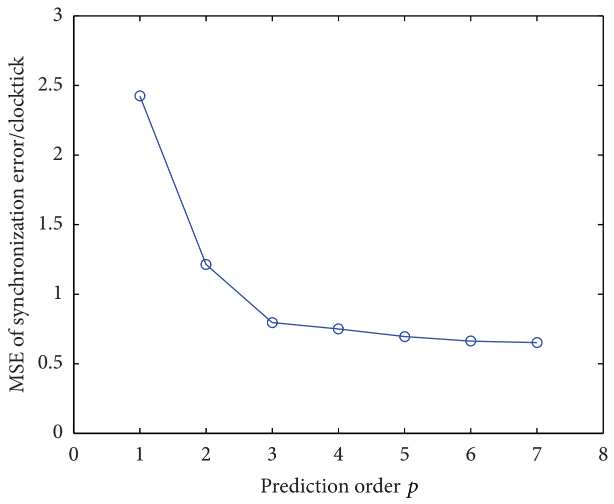

FIGURE 5: Relationship between MSE of synchronization errors E_n and prediction order p.

TABLE 1: Experimental configuration.

Synchronization model	Master-slave
Number of tests	10
Duration	24 hrs
Prediction order	3
Number of node pairs	40
Largest hops	7
Synchronization interval	2 s
Skew range	±30 ppm
Synchronization packet length	20 bytes

The experimental scenario is a multihop wireless sensor network. The default experiment configuration parameters are given in Table 1.

The experiment is based on tree topology network. Since we are more concerned with the synchronization errors between parent nodes and children nodes, we have only analyzed synchronization performance among parent nodes and their children nodes. The analysis of the synchronization errors for the whole network will be in our future work.

4.1. Prediction Order P. From analysis in Section 3.1, we know that synchronization accuracy is highly related to the prediction order p. So we explore the relationship between the MSE E_n and p by experiment. For every value of p within range 1–7, we determine E_n with each test lasting 24 hours and identify the relationship between E_n and p as shown in Figure 5.

As Figure 5 demonstrates, E_n declines quickly as p increases when p is small. When $p \geq 3$, the decrease of E_n becomes slow with the increase of p and gradually becomes trivial. Considering the limited computing capability of nodes, we have to strike for a balance between the prediction error and the computational complexity. Since E_n changes slightly as p increases when p is bigger than 3 and the computational complexity is proportional to the square of p, we set p

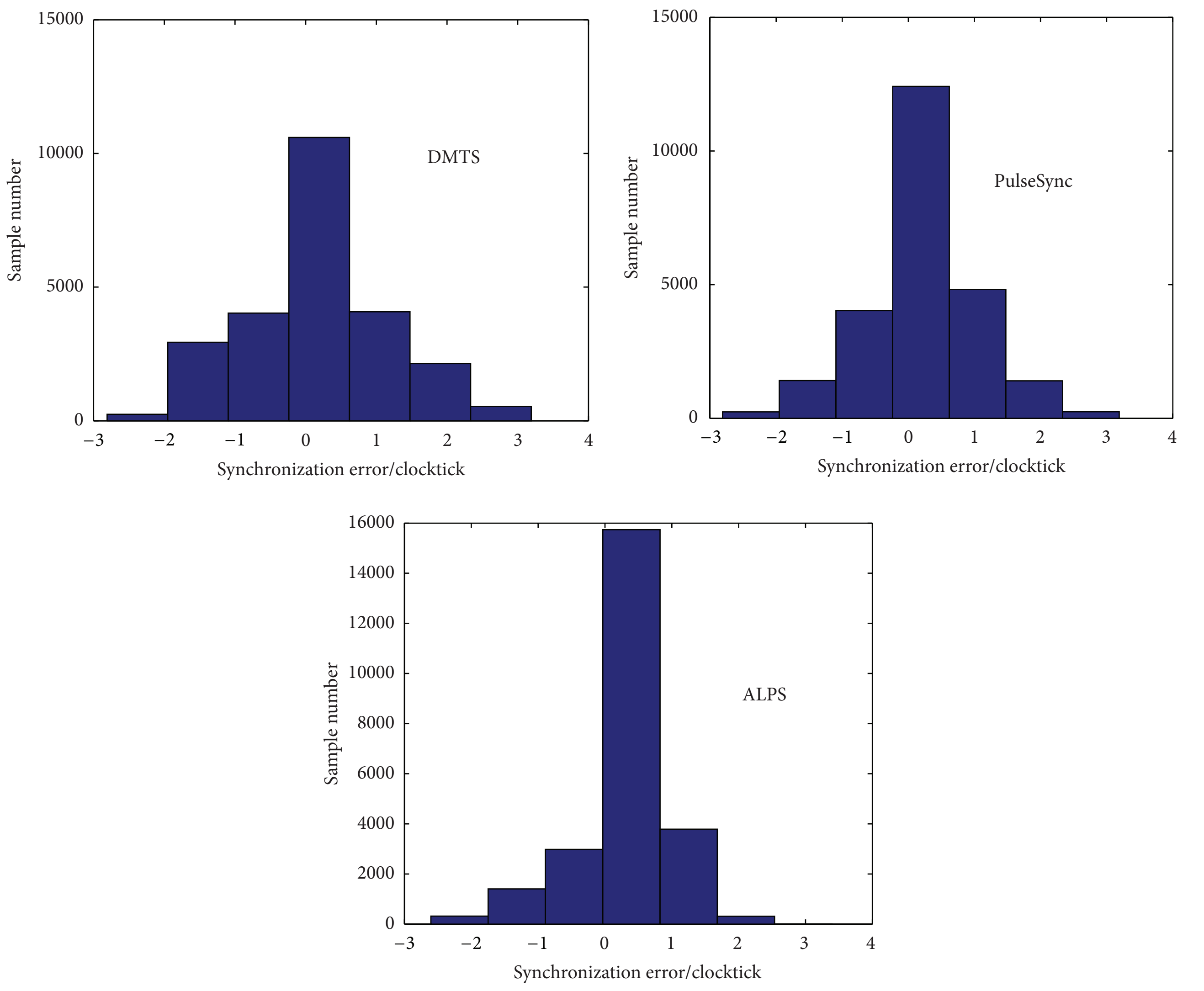

Figure 6: Histogram of synchronization errors.

as 3. When p is 3, the number of calculations and operations of ALPS is quite limited. Moreover, our hardware platform T-more sky supports complicated calculations and has an external flash with a capacity of 1024 KB for external data storage. Therefore, the ALPS is applicable in our hardware platform.

4.2. Comparison with DMTS and PulseSync. Delay measurement time synchronization (DMTS) [13] is based on the estimation of all delays involved in time synchronization message transfer path, and the delay is calculated by the length and the transmission rate of synchronization packet. PulseSync [16] is presented to reduce the global synchronization errors by making the synchronization messages disseminated as fast as possible. And the clock skew is estimated by least-squares linear regression (LSLR) on eight data points (8LR). In our experiments, we evaluate DMTS, PulseSync, and ALPS algorithms on the same platform, the Tmote-sky platform.

To compare ALPS with DMTS and PulseSync, we implemented these algorithms into the testbed in three experiments.

Table 2: Statistics of synchronization errors.

	DMTS	PulseSync	ALPS
Average error (μs)	31.34	25.26	19.33
Mean square error (μs)	40.22	31.37	23.76
Percentage of time error less than or equal to average error	45	52	63

In each experiment, the SINK node periodically broadcasts synchronization messages to the whole network, and the other nodes receive the synchronization messages and run DMTS, PulseSync, and ALPS, respectively, and calibrate their clocks to be synchronized with the SINK.

Figure 6 plots the histogram of the synchronization errors of the DMTS, PulseSync, and ALPS algorithms, and the results are also summarized in Table 2. As shown in Table 2, the MSE of ALPS declines by 41% and 24%, respectively, compared with DMTS and PulseSync. This is because DMTS

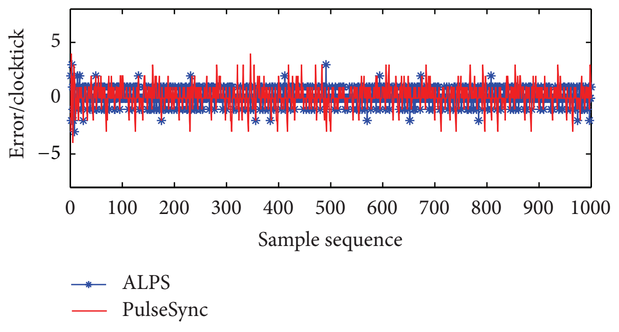

FIGURE 7: Synchronization errors of PulseSync and ALPS over time.

compensates the random synchronization delay with a fixed value, while PulseSync and ALPS use the historical data points to estimate the current synchronization error. The difference between PulseSync and ALPS is analyzed as follows. PulseSync uses eight data points (8LR) where the weights of all sample points are equal for estimation. Therefore, the degree of freedom (DOF) of PulseSync is only two (clock offset and skew) although it has eight data points. On the other hand, the prediction coefficients in ALPS are not equal and adaptively change with time (as shown in Section 3.2), so the DOF of ALPS is equal to the prediction order p. In this experiment, p is 3.

In short, PulseSync uses eight samples with the DOF as two. However, although ALPS only uses three data points, its DOF is three. Apparently, the efficiency of ALPS is higher than PulseSync.

Figure 7 depicts the synchronization errors of PulseSync and ALPS along the timeline for 1000 samples (2 s per sample). Compared with Figure 3 (without prediction), we can find that in ALPS the variance of errors is smaller, and the outliers are less and are more stable.

Through analysis on data of synchronization errors, we can get error distribution falling in different ranges. As demonstrated by Figure 8, the error distribution of ALPS is more concentrated than DMTS and PulseSync; hence the MSE of ALPS is also smaller than DMTS and PulseSync. 85% and 75% of the errors in ALPS and PulseSync fall in ±1 clocktick, respectively, whereas almost 100% of the errors in APLS fall in ±3 clocktick. This indicates that the distribution of synchronization errors in ALPS is more concentrated and falls in a smaller range (±1 clocktick) with a larger probability.

4.3. Synchronization Cost. Since the synchronization cost is highly correlated with the amount of the synchronization packets (assume the transceiver is turned off when no message is sent or received), we take the number of synchronization packets to represent the synchronization cost.

Considering the scenario where one reference node and n nodes to be synchronized are within a single broadcast domain, the synchronization cost of different schemes is given in Table 3. In RBS, K denotes the number of reference broadcasts. As K increases, the synchronization error goes down while the energy consumption goes up. According to [12],

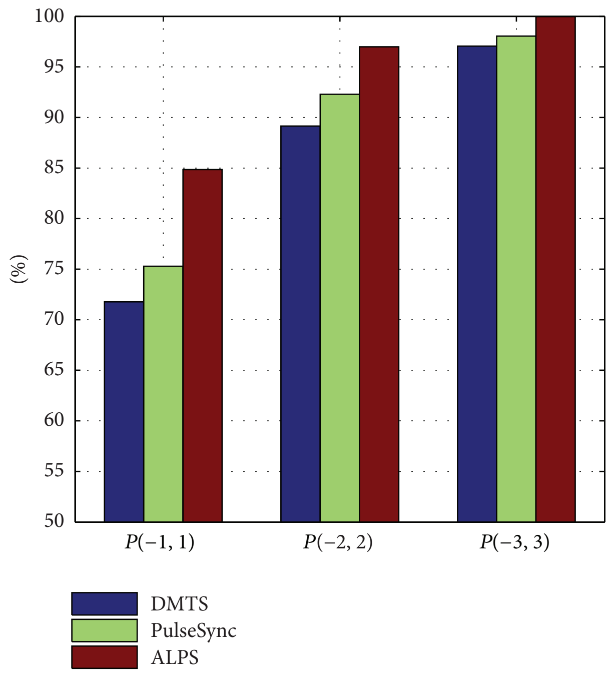

FIGURE 8: Probability of synchronization errors falling into different ranges.

TABLE 3: Comparison of synchronization cost.

	ALPS	PulseSync	DMTS	RBS	TPSN
Times of sending	n	n	n	$K(n+1)$	$2n$
Times of receiving	n	n	n	$K(n+1)$	$2n$

the average synchronization errors of RBS and TPSN are 29.13 μs and 16.9 μs, respectively. The average synchronization error of ALPS is 19.33 μs. As Table 3 demonstrates, ALPS has a big advantage in synchronization cost, which is at least half of both RBS and TPSN. Although ALPS, PulseSync, and DMTS share the same synchronization cost, as shown in Table 2, ALPS has a better performance than PulseSync and DMTS in terms of both average and MSE of synchronization errors.

We also compare PulseSync and ALPS in resource consumption. PulseSync uses LSLR to estimate the clock skew, and a typical LSLR needs a single loop with complexity $O(l)$, where l is the number of historical data points used in LSLR. However, as Section 3.3 analyzed, the complexity of ALPS is $O(p^2)$, where p is the prediction order. In our scheme, l and p are equal to 8 and 3, respectively. Compared with PulseSync, ALPS needs a little more computational complexity. This is the price ALPS has to pay for a higher performance. However, we think the tradeoff is definitely acceptable since the complexity only increases by 12.5%.

In summary, ALPS performs the same as RBS and TPSN in synchronization accuracy with a less cost and performs better than PulseSync and DMTS at a similar expense. Therefore, when both performance and cost are taken into consideration, APLS performs better than the other four synchronization schemes.

5. Conclusion

Considering the high temporal correlation of time synchronization errors, this paper proposes ALPS, a synchronization scheme based on adaptive linear prediction. Through linear prediction on the clock offset and adaptively adjusting prediction coefficients, ALPS improves synchronization accuracy without increasing synchronization costs. Experiments on the Tmote-sky platform indicate that ALPS cuts synchronization cost by at least half compared with RBS and TPSN without sacrificing accuracy and performs better than DMTS and PulseSync with the same cost. The MSE of synchronization errors in ALPS declines by 41% and 24% compared to DMTS and PulseSync, respectively.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by the National Science & Technology Pillar Program during the 12th Five-Year Plan Period (no. 2012BAJ24B01).

References

[1] Y.-C. Wu, Q. Chaudhari, and E. Serpedin, "Clock synchronization of wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 124–138, 2011.

[2] J. Zheng and Y.-C. Wu, "Joint time synchronization and localization of an unknown node in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, part 1, pp. 1309–1320, 2010.

[3] J. Wang, Q. Gao, H. Wang, Y. Yu, and M. Jin, "Time-of-flight-based radio tomography for device free localization," *IEEE Transactions on Wireless Communications*, vol. 12, no. 5, pp. 2355–2365, 2013.

[4] Z. Zhong and T. He, "Msp: multi-sequence positioning of wireless sensor nodes," in *Proceedings of the 5th ACM International Conference on Embedded Networked Sensor Systems (SenSys '07)*, pp. 15–28, Association for Computing Machinery, November 2007.

[5] J. Wang, Q. Gao, Y. Yu, P. Cheng, L. Wu, and H. Wang, "Robust device-free wireless localization based on differential RSS measurements," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 12, pp. 5943–5952, 2013.

[6] J. Wang, D. Fang, X. Chen, Z. Yang, T. Xing, and L. Cai, "LCS: compressive sensing based device-free localization for multiple targets in sensor networks," in *Proceedings of the 32nd IEEE Conference on Computer Communications (INFOCOM '13)*, pp. 145–149, Turin, Italy, April 2013.

[7] K. W. Cheung, H. C. So, W.-K. Ma, and Y. T. Chan, "Least squares algorithms for time-of-arrival-based mobile location," *IEEE Transactions on Signal Processing*, vol. 52, no. 4, pp. 1121–1128, 2004.

[8] Y. Huang, J. Benesty, G. W. Elko, and R. M. Mersereau, "Real-time passive source localization: a practical linear-correction least-squares approach," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 8, pp. 943–956, 2001.

[9] Y. Jiang, Q. Hu, and D. Yang, "Analysis of positioning error for two-dimensional location system," *Mathematical Problems in Engineering*, vol. 2013, Article ID 163958, 8 pages, 2013.

[10] D. Niculescu and B. Nath, "Ad hoc positioning system (APS) using AOA," in *Proceedings of the IEEE Societies 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM '03)*, vol. 3, pp. 1734–1743, IEEE, 2003.

[11] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 147–163, 2002.

[12] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 138–149, ACM, November 2003.

[13] S. Ping, "Delay measurement time synchronization for wireless sensor networks," Tech. Rep. IRB-TR-03-013, Intel Research Berkeley Lab, 2003.

[14] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 39–49, ACM, November 2004.

[15] Z. Zhong, P. Chen, and T. He, "On-demand time synchronization with predictable accuracy," in *Proceedings of the IEEE INFOCOM*, pp. 2480–2488, IEEE, Shanghai, China, April 2011.

[16] C. Lenzen, P. Sommer, and R. Wattenhofer, "Optimal clock synchronization in networks," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*, pp. 225–238, Berkeley, Calif, USA, November 2009.

[17] M. Leng and Y.-C. Wu, "On clock synchronization algorithms for wireless sensor networks under unknown delay," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 1, pp. 182–190, 2010.

[18] M. Leng and Y.-C. Wu, "Low-complexity maximum-likelihood estimator for clock synchronization of wireless sensor nodes under exponential delays," *IEEE Transactions on Signal Processing*, vol. 59, no. 10, pp. 4860–4870, 2011.

[19] B. R. Hamilton, X. Ma, Q. Zhao, and J. Xu, "ACES: Adaptive clock estimation and synchronization using Kalman filtering," in *Proceedings of the 14th Annual International Conference on Mobile Computing and Networking, (MobiCom '08)*, pp. 152–162, September 2008.

[20] Z. Yang, J. Pan, and L. Cai, "Adaptive clock skew estimation with interactive multi-model kalman filters for sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC '10)*, pp. 1–5, IEEE, May 2010.

[21] M. Jin, D. Fang, X. Chen, Z. Yang, C. Liu, and X. Yin, "Voltage-aware time synchronization for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2014, Article ID 285265, 13 pages, 2014.

[22] M. Leng and Y.-C. Wu, "On joint synchronization of clock offset and skew for Wireless Sensor Networks under exponential delay," in *Proceedings of the IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems (ISCAS '10)*, pp. 461–464, IEEE, June 2010.

[23] D. R. Jeske, "On maximum-likelihood estimation of clock offset," *IEEE Transactions on Communications*, vol. 53, no. 1, pp. 53–54, 2005.

[24] J. Chen, Q. Yu, Y. Zhang, H.-H. Chen, and Y. Sun, "Feedback-based clock synchronization in wireless sensor networks: a control theoretic approach," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 6, pp. 2963–2973, 2010.

[25] P. Castiglioni, "Levinson-durbin algorithm," in *Encyclopedia of Biostatistics*, John Wiley & Sons, 2005.