Delay Compensated One-Way Time Synchronization in Distributed Wireless Sensor Networks

Zihan Fang, *Student Member, IEEE*, and Yue Gao, *Senior Member, IEEE*

***Abstract*—Wireless sensor networks require accurate time synchronization to perform collaborative tasks on multiple distributed sensor nodes. However, the performance is greatly affected by the unpredictability of message delays in the synchronization process, which causes cumulative error and challenges to clock parameter estimations. We propose a delay compensated consensus Kalman-based time synchronization (DCCKTS) protocol to estimate the fluctuated delay effectively. We propose an algorithm to update all parameters in a one-way message exchange while maintaining time synchronization accuracy and energy consumption. The performance of the proposed algorithm is compared with traditional algorithms. Simulation results demonstrate that the algorithm performs better than conventional algorithms under an unknown Gaussian delay model.**

***Index Terms*—One-way time synchronization, delay measurements, Kalman filter, wireless sensor networks.**

I. Introduction

Time synchronization is fundamental for wireless sensor networks (WSNs). It is crucial that all sensor nodes run on the same timescale to handle distributed and cooperative tasks in many applications such as deterministic scheduling, data fusion, and power management [1], [2].

Compared with hierarchy-based synchronization techniques like Reference-Broadcast Synchronization (RBS) [3] and the Flooding Time Synchronization Protocol (FTSP) [4], consensus-based time synchronization techniques like [5] have attracted much attention, owing to the scalability of dynamic networks and robustness to node failure. However, accurate time synchronization obtained in [5] is under the assumption that message delays are negligible.

In practice, time synchronization in WSNs is inevitably influenced by message delays as shown in [4]. The message delays including transmission, media access, and reception process may change from time to time [3]. Such non-deterministic message delays can be magnitudes larger than the required precision [4], thus affecting the synchronization accuracy significantly. As revealed in [6]–[8], these algorithms may diverge when message delays are considered.

One way of dealing with message delays is to measure the fixed delay utilizing distributed Kalman filter, such as [9]–[11]. But they can only acquire the optimal estimation of the fixed delay and have to be realized by the two-way message exchange network. The two-way message exchange mechanism combined with the maximum likelihood method like [12], [13] is proposed as the most general approach to compensate for the fixed delay and eliminate the impact of delays in clock parameters estimation [14]. However, those two-way message exchange algorithms achieve synchronization at the expense of additional message transmissions which increases nodes' energy consumption and computational complexity [12].

Since the power resources of sensor nodes are strictly limited [12], time synchronization based on the one-way message exchange is more energy-efficient and easy-realizable because only unidirectional transmissions are required. Therefore, one-way message exchange mechanism has been widely used in practical WSNs [12]. Recently, [7], [8], [12] are presented to deal with delay compensation while still using the one-way message exchange mechanism. Unfortunately, as argued in [15], the clock offset and the message delays cannot be differentiated with only one-way messages, so the supplementary packet is required. The performance of these protocols is limited and the synchronization errors may reach the maximum fluctuation of delays because those non-deterministic delays result in poor offset estimation. Consequently, how to estimate clock skew accurately and acquire more effective offset estimation in the presence of unknown message delay fluctuation with one-way exchange is still of great interest, but very challenging.

In this letter, we propose a joint design of delay compensation and one-way synchronization, which achieves estimation of clock skew and message delays at the same time. The proposed algorithm provides better offset estimation performance under Gaussian distributed delay. The main contributions of this letter are as follows.

1) We derive a modified version of the clock model for the delay estimation scheme that performs consensus time synchronization with only one-way message to reduce consumption.

2) We propose a time synchronization algorithm to estimate clock parameters and delays based on the Kalman filter under Gaussian distributed delay. Compared with the transitional algorithms, the proposed algorithm is able to differentiate actual receiving time and fluctuated delays of non-zero mean value, hence improving synchronization accuracy in the network.

3) With the accurate delay estimation, we design an improved clock offset estimation algorithm. The simulation results demonstrate the improvement in

Manuscript received 20 June 2022; accepted 5 July 2022. Date of publication 11 July 2022; date of current version 7 October 2022. The associate editor coordinating the review of this article and approving it for publication was S. Ma. *(Corresponding author: Yue Gao.)*

Zihan Fang is with the School of Information Science and Technology, Fudan University, Shanghai 200433, China (e-mail: 19210720032@fudan.edu.cn).

Yue Gao is with the School of Computer Science, Fudan University, Shanghai 200438, China, and also with Peng Cheng Laboratory, Shenzhen 518052, China (e-mail: yue.gao@ieee.org).

Digital Object Identifier 10.1109/LWC.2022.3189744

synchronization accuracy compared with other similar works.

II. System Model

The consensus time synchronization algorithm aims to achieve an internal consensus within the network by generating a common virtual clock with skew and offset relative to the real time. And it is worth noting that the common virtual time is not necessarily the same as the real time [13].

For each node *i*, its local time $C_i(t)$ is produced by counting the hardware crystal clock, and can be modeled as a linear function of the real time *t*

$$C_i(t) = \alpha_i t + \beta_i \tag{1}$$

where α_i and β_i denote hardware skew and offset respectively.

As the real time *t* is not available to nodes, α_i and β_i cannot be obtained directly. Instead, all nodes in the network could exchange their time-related information to reach a consensus about the real time. Denote $C_v(t)$ as the consensus virtual clock, where α_v and β_v are virtual skew and offset

$$C_v(t) = \alpha_v t + \beta_v \tag{2}$$

The main goal of the consensus time synchronization is to map its local time $C_i(t)$ to the estimated virtual time and have every node estimate its clock skew $a_i(t)$ and offset $b_i(t)$ in order to track the consensus virtual time

$$L_i(t) = \frac{1}{a_i(t)} C_i(t) + b_i(t) \tag{3}$$

where the logical time $L_i(t)$ means the virtual global time estimated by node *i*.

In the one-way time synchronization, each node in the network broadcasts periodically and adjusts its $a_i(t)$ and $b_i(t)$ after receiving information from sending node. So our goal is to synchronize nodes' logical time $L_i(t)$ with only one broadcast and keep their virtual time in consensus, i.e.,

$$\forall i \in N, \lim_{t\to\infty} L_i(t) = C_v(t) = a_v t + b_v \tag{4}$$

that is $\lim_{t\to\infty} \frac{\alpha_i}{a_i(t)} = a_v$ and $\lim_{t\to\infty} \frac{\beta_i}{a_i(t)} + b_i(t) = b_v$.

Note that if node *j* broadcasts at the real time *t*, node *i* will record receiving time as $C_i(tc_i)$ at $tc_i = t + d_i(t)$ due to the exist of message delay $d_i(t)$. This means that receiving node *i* thinks that sending node *j* has $C_j(t)$ at tc_i while it is $C_j(tc_i)$ actually. As proved in [7], [8], the different record time between tc_i and *t* will bring deviation to the skew estimation of receiving node, and introduce delay fluctuation in offset estimation, leading to the limited synchronization accuracy and even divergency of network. So it is necessary to joint estimate the fluctuated delay $d_i(t)$ and the actual receiving time without delay $C_i(t)$, thus compensate clock parameters to reduce the influence from delay. The modified receiving time $C_i^+(t)$ and logical time $L_i^+(t)$ are performed as follows

$$C_i^+(t) = C_i(tc_i) - d_i(t), b_i^+(t) = b_i(tc_i) + \frac{d_i(t)}{a_i(tc_i)} \tag{5}$$

$$L_i^+(t) = \frac{1}{a_i(t)} C_i^+(t) + b_i^+(t) \tag{6}$$

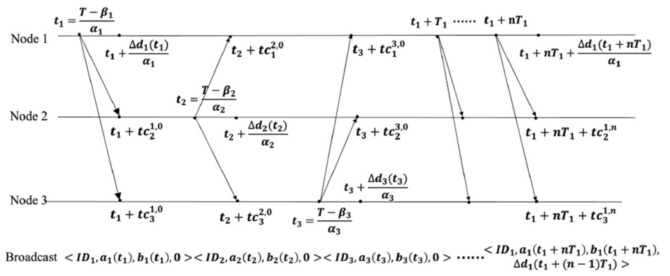
Node 1
Node 2
Node 3
$t_1 = \frac{T-\beta_1}{\alpha_1}$
$t_1 + \frac{\Delta d_1(t_1)}{\alpha_1}$
$t_2 + tc_1^{2,0}$
$t_3 + tc_1^{3,0}$
$t_1 + T_1$ ...... $t_1 + nT_1$
$t_1 + nT_1 + \frac{\Delta d_1(t_1+nT_1)}{\alpha_1}$
$t_2 = \frac{T-\beta_2}{\alpha_2}$
$t_1 + tc_2^{1,0}$
$t_2 + \frac{\Delta d_2(t_2)}{\alpha_2}$
$t_3 + tc_2^{3,0}$
$t_1 + nT_1 + tc_2^{1,n}$
$t_3 + \frac{\Delta d_3(t_3)}{\alpha_3}$
$t_1 + tc_3^{1,0}$
$t_2 + tc_3^{2,0}$
$t_3 = \frac{T-\beta_3}{\alpha_3}$
$t_1 + nT_1 + tc_3^{1,n}$
Broadcast $< ID_1, a_1(t_1), b_1(t_1), 0 >< ID_2, a_2(t_2), b_2(t_2), 0 >< ID_3, a_3(t_3), b_3(t_3), 0 >$ ...... $< ID_1, a_1(t_1+nT_1), b_1(t_1+nT_1), \Delta d_1(t_1+(n-1)T_1) >$

Fig. 1. One-way synchronization with a next round broadcast from sending node.

As a result, the receiving node *i* is able to calculate the actual offset $b_i(t)$ at time *t* and differentiate the actual receiving time $C_i(t)$, as if there is no delay in the receiving node, that is $C_i^+(t) = C_i(t)$, $\lim_{t\to\infty} L_i^+(t) = C_v(t)$.

III. Proposed DCCKTS Protocol

A. One-Way Broadcast Scheme

To realize the convergence of logical time for all nodes, the clock parameters need to be updated regularly. We assume that each node *i* broadcasts periodically with the pseudo-period *T*, then the real period is defined as $T_{\rm i} = \frac{T}{\alpha_i}$, due to the deviation from clock skew α_i.

An example of the one-way broadcast scheme between three nodes is shown in Fig. 1. For the *n*th round of communication, every node *i* takes turn to periodically broadcast its time information at the specific time t_i (such as $C_i(t_i) = nT$, that is $t_i = \frac{nT-\beta_i}{\alpha_i}$), while staying in receiving state at other time, in this case, other nodes *j* keep their clock parameters and adjust them only when receiving data at time $t_i + tc_j^i$.

For receiving node *j*, as mentioned above, the corresponding local time $C_j(t_i)$ and the fluctuated delay $d_j(t_i)$ are impossible to be jointly estimated with only one-way messages [15]. Thus the supplementary information containing the record of the current fluctuated delay at sending node *i* is introduced, and node *j* could use this transmission delay as a prior to message delays and then achieve a better delay estimation.

To provide a delay supplement without incurring excessive energy overhead, we measure the transmission delay $\Delta d_i(t_i) = C_i(t_i + \frac{\Delta d_i(t_i)}{\alpha_i}) - C_i(t_i)$ before and after sending a packet using medium access interrupt handlers, which enables capturing timing information without interactive timestamp at almost no cost. Since an extra broadcasting message would increase energy consumption, therefore, the transmission delay $\Delta d_i(t_i)$ is sent in the next broadcast at time $t_i + T_i$ together with other information. The broadcast packet is shown in Fig. 1. At the same time, the time information of node *i* at time t_i is firstly recorded at receiving node *j*, which then performs the DCCKTS algorithm after receiving the transmission delay $\Delta d_i(t_i)$ in the next round of communication.

As depicted in Fig. 1, node 1 broadcasts at time t_1, and when node 2 and 3 receive data at time $t_1 + tc_2^{1,0}$ and $t_1 + tc_3^{1,0}$, they record time information of both node 1 and themselves. After finishing sending packet, node 1 calculates the transmission delay $\Delta d_1(t_1)$ at time $t_1 + \frac{\Delta d_1(t_1)}{\alpha_1}$ and sends it at time $t_1 + T_1$. In the next round, when node 2 receives data again from node 1 at time $t_1 + T_1 + tc_2^{1,1}$, its clock parameter is adjust based on $\Delta d_1(t_1)$, $C_2(t_1 + tc_2^{1,0})$ and $C_1(t_1)$.

B. Model Establishment

Assume that node *j* is the sending node at current time *t*, and when node *i* receives the packet containing sending time $C_j(t)$, it records its receiving time $C_i(tc_i)$ at once. Since the synchronization period is constant as *T*, the periodic broadcast in node *j* corresponds to the periodic reception in node *i*. According to the local clock model in (1), the relationship between node *i* and *j* can be obtained that

$$\frac{C_i(t) - C_i(t - T_i)}{\alpha_i} = T = \frac{C_j(t) - C_j(t - T_j)}{\alpha_j} \tag{7}$$

On the other hand, based on the convergence of the logical time in (4), the skew of the common virtual clock satisfies

$$a_v = \lim_{t\to\infty} \frac{\alpha_i}{a_i(\mathrm{t})} = \lim_{t\to\infty} \frac{\alpha_j}{a_j(\mathrm{t})} \tag{8}$$

Note that $C_i^+(t) = C_i(t)$, then combine (7)(8) and analyze the stable state of the network, we have

$$\frac{C_i^+(t) - C_i^+(t - T_i)}{C_j(t) - C_j(t - T_j)} = \frac{\alpha_i}{\alpha_j} = \frac{a_i(\mathrm{t})}{a_j(\mathrm{t})} \tag{9}$$

Using the periodic broadcast *T* as the time interval between two iterations, that is, $C_j(t) - C_j(t - T_j) = T$, and the modified receiving time can be expressed as

$$C_i^+(t) = C_i^+(t - T_i) + \frac{T}{a_j(t)} a_i(t - T_i^j) \tag{10}$$

where T_i^j stands for the interval in the two adjacent reception recorded by node *i*. Since nodes only synchronize when receiving data, so $a_j(t)$ and $a_i(t - T_i^j)$ denote the latest skew estimation at node *j* and *i* respectively.

To track delay's fluctuation and well explain the relationship between local receiving time and current delay, (10) can be rewritten as follows

$$C_i^+(t) = C_i^+(t - T_i) + \left[\frac{T}{a_j(t)} - \frac{d_i(t - T_i^j)}{a_i(t - T_i^j)}\right] \cdot a_i(t - T_i^j) + d_i(t - T_i^j) \tag{11}$$

The delays in WSNs can be categorized as the fixed portion and random portion, the random portion is usually in Gaussian distribution [10], [14], [16]. Moreover, it is worth noting that the Gaussian delay model is a rough approximation of the real system [12], so $d_i(t)$ is assumed to be an independent Gaussian RV with mean μ_d and variance σ_d^2 in this letter.

In order to estimate delay accurately, supplementary information of transmission delay $\Delta d_j(t)$ from sending node *j* is necessary so as to roughly restrict the estimation around μ_d. Since we measure $\Delta d_j(t)$ with interrupt handlers and the processing time at transmitter and receiver can be considered equal [6], $\Delta d_j(t)$ represents message delays approximately. Note that we hope to estimate the current value of the fluctuated delay hence a suitable fluctuation range $w_d(t)$ is taken to describe the independence and randomness of delay. As discussed in [4], the non-determinacy of transmission and reception delay are considered equal and smaller than media access delay, so we choose variance of $w_d(t)$ following $\sigma_d^2/4$. Therefore, the estimation of delay is chosen as

$$\mathrm{d}_i(\mathrm{t}) = \Delta d_j(t) + w_d(t) \tag{12}$$

As mentioned in (8), it can be seen that the estimation of skew tends to stay stable, so it is reasonable to adopt the random walk model for the skew estimation, that is

$$a_i(t) = a_i(t - T_i^j) + w_a(t) \tag{13}$$

where $w_a(t)$ is the fluctuation of skew estimation with variance σ_a^2 and much smaller than the maximum difference between nodes' skew.

C. Kalman Synchronization

According to the description above, the clock skew $a_i(t)$, modified receiving time $C_i^+(t)$ and the fluctuated delay $d_i(t)$ are required to be estimated as $\hat{a}_i(t)$, $\hat{C}_i(t)$ and $\hat{d}_i(t)$ respectively. The Kalman state model is formulated as

$$\hat{X}_i(t) = \begin{bmatrix} \hat{a}_i(t) \\ \hat{C}_i(t) \\ \hat{d}_i(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ A_{21}(t) & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{a}_i(t - T_i^j) \\ \hat{C}_i(t - T_i) \\ \hat{d}_i(t - T_i^j) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \Delta d_j(t) \end{bmatrix} + \begin{bmatrix} \sigma_{ax}^2 & 0 & 0 \\ 0 & \sigma_{Cx}^2 & 0 \\ 0 & 0 & \sigma_{dx}^2 \end{bmatrix} = A_i(\mathrm{t})\hat{X}_i(t - T_i^j) + B_i(t) + W_i \tag{14}$$

$$A_{21}(t) = \frac{T}{\hat{a}_j(t)} - \frac{\hat{d}_i(t - T_i^j)}{\hat{a}_i(t - T_i^j)} \tag{15}$$

The measurement model is based on the one-way broadcast scheme. It is important to remark that the receiving time $C_i(tc_i)$ is recorded later by delay $d_i(t)$ in comparison with the sending time $C_j(t)$. So the observation of $C_i(tc_i)$ is

$$C_i(tc_i) = \hat{C}_i(t) + \hat{d}_i(t) \tag{16}$$

Rewrite (9) we have $a_i(t) = \frac{\alpha_i}{\alpha_j} \cdot a_j(t)$. Although the calculation of α_i based on $C_i(tc_i)$ in receiving node *i* is inaccurate influenced by the fluctuation of message delays, the observation of estimated skew $a_i(t)$ can still be formulated as

$$a_{ij}(t) = \frac{C_i(tc_i) - C_i(tc_i - T_i)}{C_j(t) - C_j(t - T_j)}$$
$$a_{oi}(t) = a_{ij}(t)\hat{a}_j(t) \tag{17}$$

Thus, the measurement equation follows as

$$Y_i(t) = \begin{bmatrix} a_{oi}(t) \\ C_i(tc_i) \end{bmatrix} = H_i\hat{X}_i(\mathrm{t}) + V_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \hat{a}_i(t) \\ \hat{C}_i(t) \\ \hat{d}_i(t) \end{bmatrix} + \begin{bmatrix} \sigma_{ay}^2 & 0 \\ 0 & \sigma_{Cy}^2 \end{bmatrix} \tag{18}$$

The Kalman filtering process can be summarized as

$$\begin{aligned} \hat{X}_i^-(t) &= A_i(\mathrm{t})\hat{X}_i(t - T_i^j) + B_i(t) \\ P_i(\mathrm{t}) &= A_i(\mathrm{t})M_i(t - T_i^j)A_i(\mathrm{t})^T + W_i \\ K_i(\mathrm{t}) &= P_i(\mathrm{t})H_i{}^T\left(V_i + H_iP_i(\mathrm{t})H_i{}^T\right)^{-1} \\ \hat{X}_i(\mathrm{t}) &= X_i^-(t) + K_i(\mathrm{t})\left(Y_i(\mathrm{t}) - H_i\hat{X}_i^-(t)\right) \\ M_i(\mathrm{t}) &= (\mathrm{I} - K_i(\mathrm{t})H_i)P_i(\mathrm{t}) \end{aligned} \tag{19}$$

Algorithm 1 Delay Compensated Consensus Kalman Based Time Synchronization

```
1:  Initialization
2:  For each node, initial Kalman matrix, and set $\hat{a}_i(0) = 1, \hat{C}_i(0) = C_i(0), \hat{d}_i(0) = 0, \hat{b}_i^+(t) = 0$.
3:  Broadcast
4:  If node $i$ satisfies $\frac{C_i(t)}{T} \in N^+$, then broadcasts $[ID_i, \hat{a}_i(t), \hat{b}_i(t), C_i(t), \Delta d_i(t - T_i)]$ to its neighbors. Compute current transmission delay $\Delta d_i(t)$ after sending.
5:  Reception
6:  If node $i$ receives a packet from node $j$ at time $tc_i$, then records its local time $C_i(tc_i)$ instantly and saves all time information in broadcast packet.
7:  If node $i$ has a historical record $\hat{C}_i(t - T_i)$, then compute observation $Y_i(t - T_i)$ according to (17)(18), otherwise set $\hat{C}_i(t - T_i) = C_i(tc_i)$.
8:  Iteration
9:  Calculate Kalman state matrix parameter $A_{21}(t)$ by (15) and set $\hat{d}_i^-(t) = \Delta d_i(t - T_i)$.
10: Run Kalman synchronization for the iterative prediction and update based on the information recorded using (19).
11: If node $i$ never receives packet from node $j$ before, set $\hat{a}_i(t) = 1, \hat{C}_i(t) = C_i(tc_i), \hat{d}_i(t) = 0$.
12: Compute global time $G_i(t - T_i)$ and estimate modified clock offset $\hat{b}_i^+(t)$ in (20) and (21) respectively.
13: Update
14: Store $\hat{a}_i(t), \hat{C}_i(t), \hat{d}_i(t), \hat{b}_i^+(t)$.
15: Set $\hat{C}_i(t - T_i) = \hat{C}_i(t), C_i(t - T_i) = C_i(t + d_i(t))$.
```

Based on the previous analysis, the logical time of each node is almost at the exact skew after the Kalman filter. Then it is only necessary to compensate offset for the possible deviation of logical time.

We take the global time $G_i(t)$ as the reference to update offset, which combines the weighted average of receiving node i and sending node j. This global time reduces over adjustment of logical time at receiving node compared with the offset compensation only based on sending node. Therefore, the algorithm speeds up the convergence while maintaining the stability of parameter estimation.

$$G_i(t) = \frac{w_i(t) \cdot L_i^+(t) + w_j(t) \cdot L_j(t)}{w_i(t) + w_j(t)} \tag{20}$$

The weight $w_i(t)$ can be the number of packets received by node i to avoid the large disturbance injected by the newly joined node to the already synchronized network. Meanwhile, the node that has not received data for a long time due to packet loss can quickly synchronize with other nodes in the network. According to (6) and (20), the offset adjustment satisfies

$$\hat{b}_i^+(t) = G_i(t) - \frac{1}{\hat{a}_i(t)} \cdot [C_i(tc_i) - \hat{d}_i(t)] \tag{21}$$

The processing procedure of the network time synchronization is described in Algorithm 1.

IV. Simulation Results

In this section, simulations based on 1000 Monte Carlo rounds in MATLAB are performed to validate the performance of the proposed DCCKTS algorithm. To display the synchronization error in the network, we consider a five nodes multi-hop ring network, in which sensor nodes are switched on randomly with a synchronization period $T = 10s$. Since the skew deviation is commonly less than 100 ppm, we assume that the value of hardware skew α_i and offset β_i of each node is randomly selected from the set [0.999 and 1.0001] and [0, 10s] respectively. We calculate synchronization error d_{time} and skew difference d_{skew} to evaluate the maximum difference of estimated virtual time and skew, i.e.,

$$d_{time} = \max_{i,j} ||L_i(t) - L_j(t)||$$

$$d_{skew} = \max_{i,j} ||\frac{\hat{a}_i(t)}{\alpha_i} - \frac{\hat{a}_j(t)}{\alpha_j}||$$

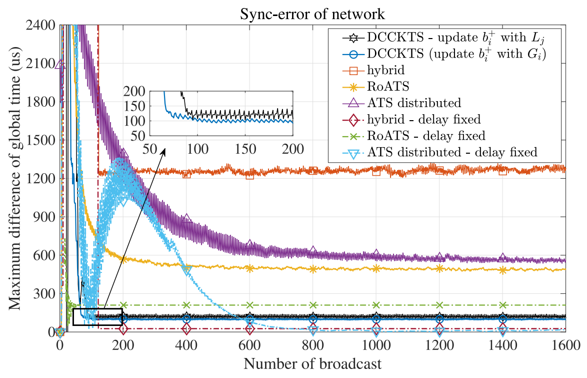

Fig. 2. Average synchronization error d_{time} of different algorithms in multi-hop network with 1000 Monte Carlo rounds.

Although several hardware platforms use the MAC layer time-stamping, which reduces the uncertain delays introduced by the transmission and the reception process, MAC layer time-stamping is not a standardized feature [6]. Hence, it is not interoperable among different hardware and physical layer protocols [6]. The proposed synchronization algorithm with message delay reflects a more realistic operation in various wireless applications.

To highlight the effect of random delays, here the fluctuated message delays follow the distribution with the mean value of $\mu_d = 2.5ms$ and variance of $\sigma_d^2 = 1ms^2$. Based on the distribution of message delays, $\sigma_{ax}^2 = 10^{-12}$ is set to balance convergence speed and accuracy, and $\sigma_{Cx}^2 = (2\sigma_{ax}T)^2, \sigma_{dx}^2 = \sigma_d^2/4, \sigma_{ay}^2 = (\sigma_d/T)^2$ are given according to (10)(12)(17). Since the observation $C_i(tc_i)$ is accurate, we set $\sigma_{Cy}^2 = 16$.

In Fig. 2, the reference of offset update as sending time $C_j(t)$ and global time $G_i(t)$ is compared. It is clear that global time reference performs better in both synchronization error and convergence speed. Then we compare the synchronization error of the proposed Kalman-based synchronization algorithm with that of the existing algorithms based on similar one-way synchronization and broadcast scheme, e.g., RoATS [8], hybrid one-way synchronization [12], ATS distributed synchronization [13] in multi-hop networks. The weight parameters $\rho_\eta, \rho_\alpha, \rho_\beta$ are set according to [8], [13] and N_1, N_2 are set to 8, 16 in [12]. These algorithms limit the fluctuation region of skew and offset estimations through the additional point-to-point

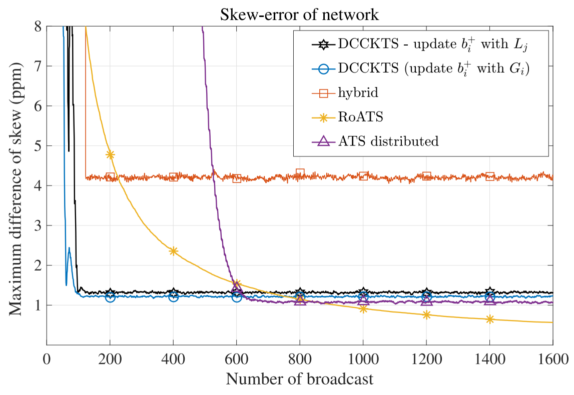

Fig. 3. Average skew difference d_{skew} of different algorithms in multi-hop network with 1000 Monte Carlo rounds.

TABLE I
COMPUTATIONAL COMPLEXITY COMPARISON

Algorithm	Parameters	Number of +/-	Number of $\times/\div$
DCCKTS algorithm	skew	$47N$	$40N$
	offset	$20N$	$17N$
Hybrid One-way Synchronization [12]	skew	$62N$	$37N$
	offset	$81N$	$41N$
RoATS algorithm [8]	skew	$20N$	$32N$
	offset	$12N$	$8N$
ATS distributed Synchronization [13]	skew	$8N$	$6N$
	offset	$5N$	$5N$

packet which helps to measure the fixed delay. As shown in Fig. 2, the proposed algorithm improves significantly in synchronization performance compared to those conventional algorithms. The reason is that DCCKTS has advantages in estimating the current fluctuated delay and differentiating it from the actual receiving time. The proposed algorithm also compensates for the fluctuation of delays in offset estimation, which affects synchronization error most significantly. Thus, it is reasonable that DCCKTS performs close to other algorithms for the case when $d_i(t)$ is fixed to μ_d.

To clarify the performance of offset estimation, the skew difference in the multi-hop network is shown in Fig. 3. As can be seen, all the algorithms achieve d_{skew} under 5ppm and perform very close after convergence, which means that the synchronization error caused by skew difference is slightly small. The difference between algorithms is less than 50us. However, the difference in d_{time} between the proposed and other algorithms is much larger in Fig. 2, which indicates the effectiveness and improvement of offset estimation in DCCKTS. Moreover, according to Fig. 2 and Fig. 3, [8] and [13] achieve synchronization convergence slowly, which increases high energy consumption for the expected accuracy.

The computational complexity of different algorithms is compared. The number of operations required for different algorithms is summarized in Table I. For those hybrid two-way synchronizations, more than one packet is needed for every iteration, while the DCCKTS algorithm only needs one broadcast, significantly saving energy consumption. [17] observed that the energy spent on sending 1 kilobyte over 100 meters is equivalent to that of executing 3 million instructions. Although DCCKTS has a slightly higher complexity in Table I, it is still energy-effective since the highly accurate synchronization can be reached with reduced additional communication overhead and fast convergence.

V. CONCLUSION

In this letter, time synchronization with fluctuated Gaussian delay in one-way message exchange for WSNs has been addressed. We designed a consensus algorithm, e.g., the DCCKTS protocol, based on the estimation of message delays. The Kalman filter was derived to eliminate the fluctuation of message delays caused by clock parameter estimations with the only one-way scheme. Simulation results have shown that DCCKTS achieved better performance and consumed less energy in comparison with conventional algorithms. The proposed DCCKTS protocol could be a strong candidate for time synchronization in distributed wireless sensor networks.

REFERENCES

[1] D. D. Geetha and N. Tabassum, "A survey on clock synchronization protocols in wireless sensor networks," in *Proc. Int. Conf. Smart Technol. Smart Nation (SmartTechCon)*, 2017, pp. 504–509.

[2] G. S. S. Chalapathi, B. Etzlinger, S. Gurunarayanan, and A. Springer, "Integrated cooperative synchronization for wireless sensor networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 3, pp. 701–704, Jun. 2019.

[3] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 147–163, Dec. 2002.

[4] M. Maroti, G. Simon, B. Kusy, and A. Ledeczi, "The flooding time synchronization protocol," in *Proc. 2nd Int. Conf. Embedded Netw. Sens. Syst.*, Nov. 2004, pp. 39–49.

[5] L. Schenato and F. Fiorentin, "Average TimeSynch: A consensus-based protocol for clock synchronization in wireless sensor networks," *Automatica*, vol. 47, no. 9, pp. 1878–1886, Sep. 2011.

[6] H. D. Soares, R. P. D. O. Guerra, and C. V. N. de Albuquerque, "FTSP+: A MAC timestamp independent flooding time synchronization protocol," in *Proc. 34th Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) Sociedade Brasileira de Computação*, 2016, pp. 820–832.

[7] J. He, P. Cheng, L. Shi, J. Chen, and Y. Sun, "Time synchronization in WSNs: A maximum-value-based consensus approach," *IEEE Trans. Autom. Control*, vol. 59, no. 3, pp. 660–675, Mar. 2014.

[8] E. Garone, A. Gasparri, and F. Lamonaca, "Clock synchronization protocol for wireless sensor networks with bounded communication delays," *Automatica*, vol. 59, pp. 60–72, Sep. 2015.

[9] B. Luo and Y. C. Wu, "Distributed clock parameters tracking in wireless sensor network," *IEEE Trans. Wireless Commun.*, vol. 12, no. 12, pp. 6464–6475, Dec. 2013.

[10] F. Kirsch and M. Vossiek, "İDistributed Kalman filter for precise and robust clock synchronization in wireless networks," in *Proc. IEEE Radio Wireless Symp.*, 2009, pp. 482–485.

[11] H. Wang, R. Lu, Z. Peng, and M. Li, "Timestamp-free clock parameters tracking using extended Kalman filtering in wireless sensor networks," *IEEE Trans. Commun.*, vol. 69, no. 10, pp. 6926–6938, Oct. 2021.

[12] H. Wang, P. Gong, F. Yu, and M. Li, "Clock offset and skew estimation using hybrid one-way message dissemination and two-way timestamp free synchronization in wireless sensor networks," *IEEE Commun. Lett.*, vol. 24, no. 12, pp. 2893–2897, Dec. 2020.

[13] R. Yang and Y. Jiang, "Consensus-based clock synchronization for wide area networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2021, pp. 1–6.

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

[15] N. M. Freris, S. R. Graham, and P. R. Kumar, "Fundamental limits on synchronizing clocks over networks," *IEEE Trans. Autom. Control*, vol. 56, no. 6, pp. 1352–1364, Jun. 2011.

[16] S. Ping, *Delay Measurement Time Synchronization for Wireless Sensor Networks*, vol. 6, Intel Res. Berkeley Lab, Berkeley, CA, USA, Jun. 2003, pp. 1–10.

[17] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, no. 5, pp. 51–58, 2000.