ICIC Express Letters
Part B: Applications
Volume **7**, Number **1**, January **2016**

 ISSN 2185-2766
pp. 135–141

STABLE TIME SYNCHRONIZATION FOR WIRELESS SENSOR NETWORK BY ADAPTIVE VALUE TRACKING

Bing Hu and Zhixin Sun

Key Laboratory of Broadband Wireless Communication and Sensor Network Technology
Nanjing University of Posts and Telecommunications
No. 66, New Mofan Road, Nanjing 210003, P. R. China
hubing_2009@163.com; sunzx@njupt.edu.cn

Received July 2015; accepted October 2015

Abstract. *A desirable time synchronization protocol based on flooding in wireless sensor networks (WSNs) should neither demand fast propagation of latest time information nor keep track of the neighboring nodes. Moreover, such a protocol is strictly required to have a stable and energy efficient synchronization process. In this paper, we propose a stable time synchronization algorithm by adaptive value tracking that has smooth convergence process and low energy consumption. By considering both logical clock value and average rate of logical clock, each sensor node keeps synchronization with the rate of reference clock through successive feedbacks, and then adjusts clock offset of the node, and the network-wide synchronization is established fast without demanding fast propagation of latest time information and keeping track of the neighboring nodes. From our example comparison, we observed that the proposed algorithm provides stable time synchronization compared to the time synchronization algorithm by adaptive value tracking.*
Keywords: Wireless sensor network, Time synchronization, Adaptive value tracking

1. **Introduction.** Time synchronization is an essential condition for many applications in wireless sensor networks. Each node in wireless sensor network is equipped with a read-only hardware clock which is generated by quartz crystal oscillator. Unfortunately, due to these unstable quartz crystal oscillators, the frequency of hardware clocks often drifts away from the rated value and the drift is different from each other. This situation leads to invalid operation of many applications in wireless sensor network. In order to achieve a common notion of time, each node maintains a logical clock which is realized by software to express the synchronized time within the network. The purpose of time synchronization is to minimize the differences between the logical clocks of nodes (clock skew) with a stable and efficient way.

There are many theoretical studies in the literature that focus on establishing a relationship between the hardware clocks of nodes and that of reference node to predict future clock values of reference node without communicating frequently [1]. The main techniques employed for this establishment are the method of least-squares [2] and occasionally distributed agreement [3]. However, those methods either demand fast propagation of latest time information or keep track of the neighboring nodes, which increase the communication overhead as well as memory use. To reduce the energy consumption, a novel mechanism that employs adaptive value tracking [4] to realize time synchronization in WSN is proposed in [5] and [6].

However, there are also some drawbacks in those papers. The heart of [5] and [6] is adjusting the rate of the nodes' logical clock through successive feedbacks which is generated by a computationally light adaptive value tracking (AVT) technique. Both of them have a common drawback, at the initial synchronization period, the rates and the values of the logical clocks often have large skews, when sensor nodes receive the logical clocks values with big error from unsynchronized sensor nodes, wrong feedbacks will be

sent to their AVTs. Hence, before the convergence, the rate multiplier values fluctuating a lot, and more times to adjust the progress rate of logical clock result in more energy consumption.

Our objective in this paper is to mitigate the fluctuation of the rate multiplier, which is introduced by the method of adaptive value tracking. For this purpose, we focus on the algorithm in [6], and modify this algorithm by considering both the clock skew and the average rate of the logical clocks to adjust the progress rate of the logical clocks. Thanks to this improvements, we obtain a more stable synchronization process, compared to [6], with fewest times to change the rate multiplier, which is compared to [5].

The remainder of this paper is organized as follows. In Section 2, we present our system model. We describe the time synchronization protocol by adaptive value tracking in Section 3 and introduce our algorithm in Section 4. The evaluation of our algorithm is described in Section 5. Finally, we present our conclusions and future work in Section 6.

2. **System Model.** In this section, we denote a real sensor network as a graph $G = \{V, E\}$ with a vertex set $V = \{1, \ldots, N\}$ and an edge set $E \subseteq V \times V$ that represent a set of sensor nodes and the bidirectional communication links between the nodes inside the wireless broadcast region of each other, respectively. The nodes that are directly connected with node $u \in V$ are referred to as the neighbors of that node and expressed as $N_u = \{v \in V \,|\{u, v\} \in E\}$. In order to simplify our analysis in the rest of the study, we assume that communication link is reliable and the network is static. Hence, a message sent by a node $u \in V$ is received by all of its neighbors N_u exactly.

We assume that each node is equipped with a read-only hardware clock suffering from clock drift. We denote the hardware clock of any sensor node $u \in V$ as $H_u(\ )$ and the reading at any real time t as

$$H_u(t) = \int_0^t h_u(\tau) d\tau$$

$h_u(\tau)$ represents the rate of the hardware clock at time τ. Since the frequencies of the crystal oscillators have a bounded drift, we assume that the rate of the hardware clock H_u at any time t has a bounded drift as $1 - \varepsilon \leq h_u(t) \leq 1 + \varepsilon$ where ε is a constant and satisfies $0 < \varepsilon << 1$.

Since the hardware clocks suffer from clock drift and cannot be modified, each node $u \in V$ maintains a logical clock $L_u(\ )$ which is a notion of global time. The value of the logical clock at time t is calculated as

$$L_u(t) = L_u(t_0) + l_u(t_0) \int_{t_0}^{t} h_u(\tau)\, d\tau = L_u(t_0) + l_u(t_0)\,(H_u(t) - H_u(t_0))$$

where t_0 is the latest time that a node receives a recent global time information and updates the progress rate and the offset of its logical clock. We define the average rate of the logical clock as $\frac{L_u(t) - L_u(t_0)}{t - t_0}$, which is represented by $\overline{l_u(t)}$. The progress rate of the logical clock at time t is denoted as

$$l_u(t) = \frac{dL_u}{dt}(t) = l_{ur}(t_0)\, h_u(t)$$

where $l_{ur}(t_0)$ denotes the rate multiplier which is calculated at the time t_0.

3. **Time Synchronization by Adaptive Value Tracking.** In this section, we summarize the time synchronization algorithms by adaptive value tracking in the literature and present their drawbacks. The heart of these algorithms is adjusting the rate multiplier to synchronize the rate of the logical clock for the sensor nodes through an adaptive value tracking technology.

A. Self-Organizing Time Synchronization Protocol

The protocol introduced in [5] works in a fully distributed self-adaptive manner, namely self-organizing time synchronization protocol (STSP), which provides time synchronization between sensor nodes by keeping track of the neighboring nodes without requiring memory repository. Every nodes in the network maintain an adaptive value tracker avt_u, which is used for searching and tracking the rate of the logical clock of the sender and the value represented by avt_u is the rate multiplier $l_{ur}(t)$ of the logical clock $L_u(t)$ at any time t. In this algorithm, sensor node transmits its time information, i.e., the value of its logical clock $< L_v >$, to its neighbors. Upon receiving time information from $v \in N_u$, node u adjusts its value of avt_u through successive feedbacks to synchronize the progress rate of the logical clock L_u. The pseudo-code of STSP is presented in Algorithm 1.

Algorithm 1 STSP pseudo-code for node u

Upon receiving $< L_v >$ *from* $v \in N_u$
  $skew \leftarrow L_v - L_u$
  if $|skew| \leq MaxSkew$ *then*
    if $skew > 0$ *then*
      $avt_u.adjust(f \uparrow)$
    else if $skew < 0$ *then*
      $avt_u.adjust(f \downarrow)$
    else $avt_u.adjust(f \approx)$ *endif*
  endif
  $L_u \leftarrow L_u + skew/2$

From Algorithm 1, node u performs calculation and adjustment after receiving the time information from its neighboring nodes, and then broadcasts its estimate value of logical clock skew to its neighboring nodes. It is considered in [5], if the skew is positive, the progress rate of the neighboring clock is faster, and a feedback about increasing l_{ur} is sent to avt_u. Considering this situation, if the average rate of logical clock $\overline{l_u}$ is greater than the corresponding value of its neighboring nodes, a greater value of $\overline{l_u}$ will be introduced due to increasing l_{ur}, which will cause feedback fluctuations and increase the frequency to adjust the rate of multiplier.

B. Adaptive Value Tracking Synchronization Protocol

The main idea of Algorithm 2 is similar to Algorithm 1, both of them are trying to synchronize the rate of clock by the technology of adaptive value tracking, and unlike STSP, the protocol proposed in [6] works in a flooding manner. In adaptive value tracking synchronization protocol (AVTS), a reference node floods its time information into the network, and each sensor node calculates its clock skew and adjusts its rate with respect to the reference clock. Compared to STSP, another variable is defined, seq_u, to store the latest sequence number received from the reference node.

The pseudo-code of AVTS is described in Algorithm 2. Whenever a synchronization message with a larger seq_u is received, it means a new synchronization process is beginning. When node u receives a new synchronization message, clock skew, which is used to adjust its logical clock rate, is calculated by subtracting the received logical clock from the value

Algorithm 2 AVTS pseudo-code for node u

Upon receiving $< L_v, seq_v >$ *such that* $seq_u < seq_v$
  $skew \leftarrow L_u - L_v$
  if $skew > \delta$ *then* $avt_u.adjust(f \downarrow)$
  else if $skew < -\delta$ *then* $avt_u.adjust(f \uparrow)$
  else $avt_u.adjust(f \approx)$ *endif*
  $L_u \leftarrow L_v$
  $seq_u \leftarrow seq_v$

of its logical clock. If the skew is greater than a predefined tolerance δ, it means the clock skew between its logical clock and the reference clock is beyond the requirement of synchronization accuracy, then node u sends a decrease feedback $f \downarrow$ to avt_u to inform that the logical clock needs to progress at a lower rate. Similarly, if the skew is smaller than tolerance δ, an increase feedback $f \uparrow$ is sent to avt_u to increase its logical clock rate. If the skew is within tolerance bounds, it means the clock skew between its logical clock and the reference clock satisfies the requirement of synchronization accuracy, and then a good feedback $f \approx$ is sent to avt_u. After adjusting the progress rate of its logical clock through the successive feedbacks, node u updates its logical clock value to the received logical clock value. Finally, update the sequence number. Since the firstly received message can be considered as carrying the most up-to-date estimate of the reference node, node u discards other messages in that synchronization process. However, it is easy to find that, at the beginning of a synchronization process, due to the big skews of the logical clocks, wrong feedback is sent to their avt_u, and a fluctuation of the rate multiplier is generated before the synchronization.

4. **Stable Time Synchronization by Adaptive Value Tracking.** On the basis of Algorithm 1 and Algorithm 2, a stable and energy efficient algorithm by adaptive value tracking is presented in this section. Similar to the algorithm mentioned in [5] and [6], each node in Algorithm 3 maintains an adaptive value tracker, i.e., avt_u, which is searching and tracking the rate of the clock of the reference node through successive feedbacks. However, unlike Algorithm 1 and Algorithm 2, the successive feedbacks are obtained by considering both logical clock value and average rate of logical clock. This guarantees a more stable synchronization process, compared to Algorithm 2, with the fewest times to change the rate multiplier, which is compared to Algorithm 1.

The pseudo-code is described in Algorithm 3. Whenever a synchronization message with a larger seq_u is received, it means a new synchronization process is beginning (Algorithm 3, line 1). Therefore, the firstly received synchronization message can be regarded as the latest time information of the reference node. In order to adjust the rate of the logical clock, the received node, i.e., node u, calculates the clock skew by subtracting the logical clock which is received from node v (Algorithm 3, line 2). In the situation that skew is greater than a predefined tolerance δ (Algorithm 3, line 3), if the average rate of the logical clock of node u is equal or greater than node v's, node u sends a message about decreasing the rate multiplier $f \downarrow$ to avt_u, to reduce the progress rate of the logical clock

Algorithm 3 stable time synchronization algorithm for node u

1: $\square$ *Upon receiving* $< L_v, \overline{l_v}, seq_v >$ *such that* $seq_u < seq_v$
2: $skew \leftarrow L_u - L_v$
3: *if* $skew > \delta$
4: *if* $\overline{l_u} \geq \overline{l_v}$ *then* $avt_u.adjust(f \downarrow)$
5: *else* $avt_u.adjust(f \approx)$
6: *else if* $skew < -\delta$
7: *if* $\overline{l_u} \leq \overline{l_v}$ *then* $avt_u.adjust(f \uparrow)$
8: *else* $avt_u.adjust(f \approx)$
9: *else*
10: *if* $\overline{l_u} > \overline{l_v}$ *then* $avt_u.adjust(f \downarrow)$
11: *else if* $\overline{l_u} < \overline{l_v}$ *then* $avt_u.adjust(f \uparrow)$
12: *else* $avt_u.adjust(f \approx)$
13: *endif*
14: $L_u \leftarrow L_u - \left\lceil skew/2 \right\rceil$
15: $seq_u \leftarrow seq_v$

(Algorithm 3, line 4). While, if the average rate of the logical clock of node u is less than node v's, node u sends a good feedback to avt_u (Algorithm 3, line 5). Similarly, In the situation that skew is smaller than $-\delta$ (Algorithm 3, line 6), if the average rate of the logical clock of node u is less than or equal to node v's, node u sends a message about increasing the rate multiplier $f \uparrow$ to avt_u, to improve the progress rate of the logical clock (Algorithm 3, line 7). While, if the average rate of the logical clock of node u is greater than node v's, node u sends a good feedback to avt_u (Algorithm 3, line 8). In the situation that skew is in the limit of $[-\delta, \delta]$, if the average rate of the logical clock of node u is greater than node v's, node u sends a message about decreasing the rate multiplier $f \downarrow$ to avt_u, to reduce the progress rate of the logical clock (Algorithm 3, line 10). If the average rate of the logical clock of node u is less than node v's, node u sends a message about increasing the rate multiplier $f \uparrow$ to avt_u, to improve the progress rate of the logical clock (Algorithm 3, line 11). Otherwise, node u sends a good feedback to avt_u (Algorithm 3, line 12). After completing the above adjustment, node u updates its logical clock value by subtracting the half of the skew (Algorithm 3, line 14). Finally, update the sequence number (Algorithm 3, line 15).

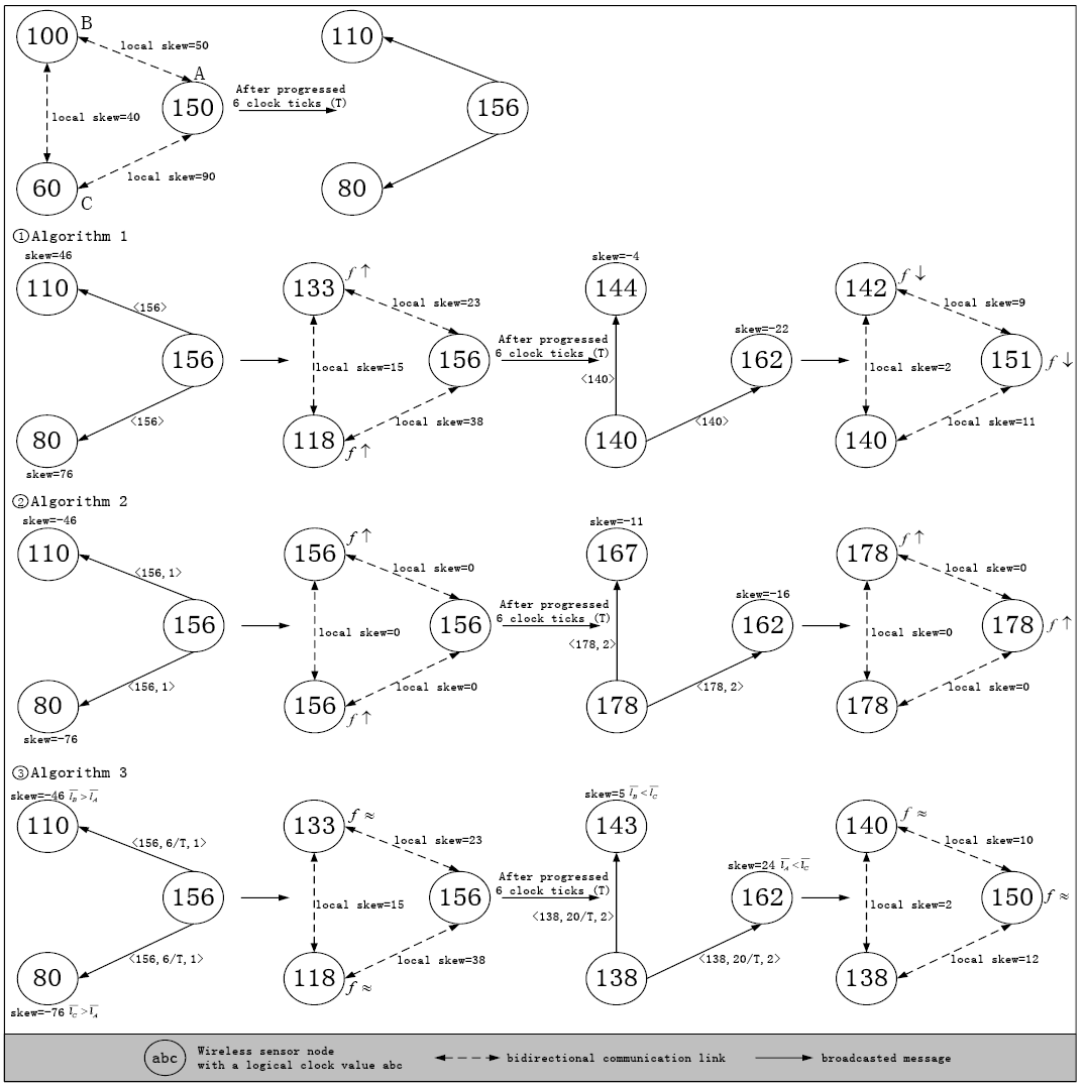

FIGURE 1. A sample execution of Algorithm 1, Algorithm 2 and Algorithm 3 on a network of 3 sensor nodes where each sensor has 2 neighbors respectively

5. **Example Comparison.** According to the sample presented in [5], Figure 1 presents the execution process of Algorithm 1, Algorithm 2 and Algorithm 3 respectively. The network for example includes only three nodes A, B and C. Initially, the logical clock values of the nodes are equal to the values of their hardware clock and the progress rates of logical clocks are the same as their hardware clock rates. We assume that the progress rate of logical clock is approximately equal to the average rate of logical clock in a small period of time. After 6 clock ticks of hardware clock, a clock tick is denoted as T, and node A broadcasts its synchronization information to its neighbors. It is easily found that the average logical clock rates of nodes B and C are both faster than node A, but the values of nodes B and C are both less than node A. Both Algorithm 1 and Algorithm 2 consider that the greater logical clock value is, the faster progress rate of logical clock is. However, the value of node A is greater than the value of node B while node A is progressing at a lower speed compared to node B. Since the received value of logical clock is greater than their logical clock values, in Algorithm 1 and Algorithm 2, the neighbors increase the progress rate of their logical clocks, while the received average rate of logical clock is slower than their average logical clock rates, so it is not changed in Algorithm 3.

After another 6 clock ticks of hardware clock, node C broadcasts its synchronization information to its neighbors. It can be observed from Algorithm 1 that the neighbors slow down the speed of logical clocks by sending a feedback to their $avts$, because the received value of logical clock is less than their logical clock values. Similarly in Algorithm 2, because the received value of logical clock is greater than their logical clock values, they increase the speed of their logical clocks by sending a feedback to their $avts$. While, in Algorithm 3, the received value of logical clock is less than their logical clock values, but the received average rate of logical clock is faster than their average logical clock rates, so the neighbors send a good feedback to their $avts$. Compared with Algorithm 1, Algorithm 3 maintains approximate convergence rate with fewer times to adjust the progress rate of logical clock, and without generating a fluctuation of the rate multiplier at the beginning of a synchronization process, which is introduced in Algorithm 2.

6. **Conclusions.** In this study, we proposed a stable time synchronization algorithm by adaptive value tracking that has smooth convergence process and low energy consumption. The algorithm maintains convergence with fewer times to adjust the progress rate of logical clock, and without generating a fluctuation of the rate multiplier at the beginning of a synchronization process, which is obtained by considering both logical clock value and average rate of logical clock. We gave an example to illustrate the stability and low energy consumption of the algorithm. A proof of the protocol's convergence and a numerical analysis of the protocol's computational overhead should be done and we leave these issues as future work.

Acknowledgment. This paper was supported by the National Natural Science Foundation of China (No. 61170276, 61373135); Project for Production Study and Research of Jiangsu Province (Grant No. BY2013011); Science and Technology Enterprises Innovation Fund Project of Jiangsu Province (Grant No. BC2013027); Key University Science Research Project of Jiangsu Province (Grant No. 12KJA520003); Natural Science Foundation of Jiangsu Province of China (Grant No. BK20140883); Research Innovation Program for College Graduates of Jiangsu Province (Grant No. KYLX15_0839).

REFERENCES

[1] K. S. Yildirim and A. Kantarci, Drift estimation using pairwise slope with minimum variance in wireless sensor networks, *Ad Hoc Networks*, pp.765-777, 2013.

[2] M. Maróti, B. Kusy, G. Simon and Á. Lédeczi, The flooding time synchronization protocol, *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems*, 2004.

[3] K. S. Yildirim and A. Kantarci, Time synchronization based on slow-flooding in wireless sensor networks, *IEEE Trans. Parallel and Distributed Systems*, pp.244-253, 2014.

[4] S. Lemouzy, V. Camps and P. Glize, Principles and properties of a mas learning algorithm: A comparison with standard learning algorithms applied to implicit feedback assessment, *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 2011.

[5] O. Gürcan and K. S. Yildirim, Self-organizing time synchronization of wireless sensor networks with adaptive value trackers, *Proc. of the 7th IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, 2013.

[6] K. S. Yildirim and O. Gürcan, Efficient time synchronization in a wireless sensor network by adaptive value tracking, *IEEE Trans. Wireless Communications*, pp.3650-3664, 2014.