Sensors & Transducers, Vol. 199, Issue 4, April 2016, pp. 29-41

Sensors & Transducers

http://www.sensorsportal.com

Less Clock Granularity Brings More Accurate Synchronization

Xiaoyuan MA, Weisheng TANG, Jianming WEI, Jun HUANG and Bo ZHANG

Shanghai Advanced Research Institute, Chinese Academy of Sciences,
No. 99, Haike Road, 201210, Shanghai, China
Tel.: +86-21-20350857, fax: +86-21-203250990
E-mail: maxy@sari.ac.cn, tangws@sari.ac.cn, wjm@sari.ac.cn, huangj@sari.ac.cn, zhangaigh@gmail.com

Received: 28 February 2016 /Accepted: 5 April 2016 /Published: 30 April 2016

Abstract: Nowadays clock synchronization has become crucial in wireless sensor network (WSN), in particular for those scenarios where the common reference time is vital for applications. The existing approaches often suffer from synchronizing errors since the effect of clock granularity is overlooked. In this paper, a novel algorithm is proposed on the basis of PulseSync algorithm. Compared with PulseSync, the improvement of the presented algorithm is twofold: First, clock discreteness is abated via applying filter method. Second, for reducing the deviation of estimated clock skew through several hops, relative logical clock rate is straightly delivered. Our approach, along with the deeper analyses, is further validated with EXP5438 platform using Contiki operating system in Cooja simulator. The result illustrates that the proposed algorithm outperforms PulseSync in synchronization accuracy. *Copyright © 2016 IFSA Publishing, S. L.*

Keywords: Clock synchronization, WSN, Clock granularity, PulseSync.

1. Introduction

[1] Recently, WSN has been applied in a wide range of fields including environment monitoring, combustible gas or gas density measurement [1-2], object detecting, military supervisory, precision agriculture and so forth [3]. Its high popularity is due to great boosts in related technologies such as electronics, Micro-electromechanical system (MEMS), wireless communication, etc. Simply put, WSN is essentially a distributed network system where a huge number of nodes are densely deployed and communicate with each other via multihops mechanism.

One of the most key issues in WSN is time synchronization which plays a very important role in a bunch of related technologies from node energy management mechanism to Media Access Control (MAC) based on time slots [4]. Abundant applications are also highly dependent on time synchronization in which a common time scalar could be obtained by nodes communicating with each other and adjusted by themselves. Therefore, many researches are focused on the aspect of WSN time synchronization in these years.

[1] This article is an extended version of our SENSORNETS 2016 paper [22].

Time synchronization in WSN is confronted with greater challenges than those center-based time synchronization systems such as GPS and Network Time Protocol-based (NTP) [5]. Above all, nodes in WSN play two roles: sender and receiver, which means that each node brings its own time into correspondence with its predecessor and meanwhile delivers or broadcasts the reference time to successors. Secondly, the node's process capability gets limited in consideration of the need of low cost and power consumptions in WSN applications. Lastly, time synchronization also need to take account into mobility, crystal frequency drift caused by environmental factors and other constraints.

This paper proposed an improved time synchronization algorithm which originates from PulseSync algorithm [6-7]. The proposed algorithm is superior to PulseSync in that the impact of clock granularity has been taken into account to deal with synchronization error induced by multihops of communication. Similar to our method, Virtual High-resolution Time (VHT) [8] also is oriented towards the issue of the clock granularity. But different from VHT, no extra hardware device/resource is required by ours.

Usually, the skew estimation is affected by the clock granularity in PulseSync and some other traditional WSN synchronization algorithms or protocols. To overcome the problem, two main contributions are achieved in this paper:

Firstly, statistic filter method is imposed on relative logical clock rate to abate the impact of discreteness of clock value.

Owing to the space distance among adjacent nodes is not far, vicinage nodes are assumed to be in the same circumstance and those crystal drift is gradual and continuous. Statistical filter such as Kalman filter is invoked to make the deviation of the relative logical clock rate less.

Secondly, the relative logical clock rate besides the clock value is delivered straightly to mitigate the deviation of the estimated clock skew.

The approach that estimated clock skew computed with clock value in common algorithms will be affected by the clock granularity naturally. On the contrary, relative logical clock rate can be expressed more precisely in the same bit width and the deviation of estimated clock skew will decrease as a result.

The organization of this paper is as follows: the related researches will be reviewed in Section 2. In Section 3, we shall describe PulseSync concisely and deep into the clock granularity effects. The improved method is proposed in Section 4. In Section 5, the results of simulation and some remarks are provided. We draw conclusion in Section 6.

2. Related Works

In the past decades, there have been numerous researches on WSN time synchronization protocol.

Timing-sync Protocol for Sensor Networks (TPSN) [9] is a classical sender-receiver synchronization (SRS) method. The synchronizing packet containing timestamps needs to be exchanged between two nodes in TPSN. One category of the criticisms about TPSN is neglecting the estimation of clock skew, and the other is its requirement of frequent communication. In order to overcome the first sort of drawback, TINY-SYNC/MINI-SYNC [10] was raised where the clock skew estimation was taken into consideration. However, the second of the issues was solved by another kind of synchronization method named receiver-receiver synchronization (RRS). Reference Broadcast Synchronization (RBS) [11] is one representative case where two receivers get timing packet from one broadcast beacon. Once obtained the packet, receivers record timestamps in accordance with their local clocks. Furthermore, the two receivers also communicate with each other for calculating the relative skew using least-squares linear regression. The great advantage of RBS is its capability of eliminating the time of the transmission and the media access. Although its success in time synchronization, RBS still gets stuck in tackling with those non-deterministic time delays in transmission, media access and receiving procedure. Concerned with the above problems, M. Maroti, *et al.* put forward flooding Time Synchronization Protocol (FTSP) [12] in which the operation of timestamping is moved down to the MAC layer and results in reduction of non-deterministic access time. In addition, FTSP decreases the communication frequency and makes that less than RBS.

Nowadays, researchers have paid more attentions to alleviating time synchronization errors in real network, especially in multihops communication scenario. They observed that neighbor nodes in practical WSN may communicate at short intervals or collaborate on executing a common task while the distant nodes seldom exchange the information. Motivated by the phenomenon, numerous improved approaches have been raised. For instance, Nancy Lynch, *et al.* proposed Gradient Clock Synchronization (GCS) theoretically in [13]. On the basis of GCS, Philipp Sommer, *et al.* implemented the Grandient Time Synchronization Protocol (GTSP) [14] where the estimated logical clock skew of each node tends to be a constant under the circumstance of strong connection. In order to synchronize with the neighbor nodes precisely in GTSP, each node will increase their logical clock skew by averaging skew among neighboring clocks. The fully distributed synchronization algorithms, e.g. GTSP, could be an approach to solving the problem of synchronization error accumulation in multi-hop network via achieving less time error in neighbor nodes, yet GTSP yields the more global time error.

Unlike GTSP, Lenzen, *et al.* [6] proposed an alternative to lessen time synchronization errors resulting from multihops. In [6], time synchronization error was analyzed. More importantly, the authors pointed out the reason why FTSP yields time synchronization error via multihops and then proposed PulseSync algorithm. The essential idea of PulseSync

is to align the delivery of synchronization packets in each node. This approach increased the accuracy between two nodes that are not adjacent by minimizing the estimated error of multihops-induced skew.

After PulseSync, there also have been significant advances in clock synchronization, especially from signal processing perspectives [15-17]. But most of them disregarded the clock granularity that may be principal factor of the error sometimes except that the Virtual High-resolution Time (VHT) was presented in [8] to refine the clock granularity with one extra hardware interrupt or logical device, e.g. FPGA. Nevertheless, the hardware interrupt resources on the MCU chip is inadequate generally and it is almost impossible to be equipped with FPGA in nodes in many rigor applications due to the limitation of power consumption and physical size. Thus, it is quite necessary to regard the clock value discreteness without any extra hardware resources and the impact of clock granularity will be analyzed in Section 3.

3. Analysis

In this part, the models and several notations used in this paper will be introduced in 3.1. Then, we shall analyze the multihops-produced error because of the clock granularity in 3.2.

3.1. Models

Individual Node

Typical clock on a WSN node consists of a crystal oscillator and a counter. The counter will decrease when the crystal oscillator oscillates once. An interrupt will be triggered and the counter resets when the counter decrease to zero. The logical clock (another clock) increases in the interrupt service routines (ISR). The application programs on WSN node read and utilize the logical clock value through the application programming interface (API).

Given the rate of crystal oscillator $h(t)$, the hardware clock $H^i(t)$ of Node i is computed as follows:

$$H^i(t) = \int_{t_0}^{t} h^i(\tau)d\tau + \Phi^i(t_0), \qquad (1)$$

where $\Phi^i(t_0)$ is the hardware clock offset at time t_0.

As shown in Fig. 1, ideally, $h^i(t)$ ought to be constant whose value is one (like the slope of the dotted line). But practical crystals will not be same exactly and many outside elements such as environmental temperature, power supply voltage and aging [18] probably cause the frequency drift, so $h^i(t)$ is not an constant, i.e.,

$$h^i(t) \in [1-\rho, 1+\rho], \qquad (2)$$

where ρ is the maximum frequency tolerance of the crystal that often can be referred to the related datasheet. Normally, the range of ρ is 1 to 100 ppm.

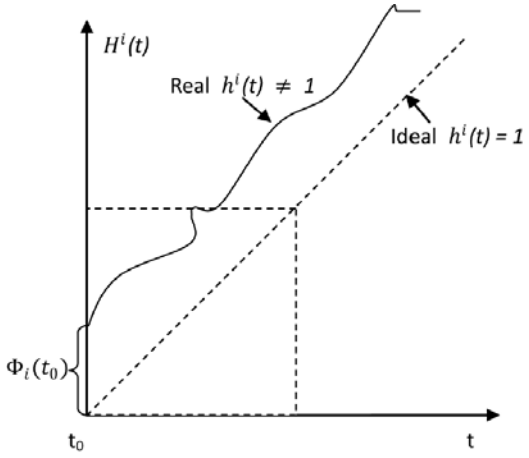

Fig. 1. Rate of crystal oscillator.

WSN node is unable to adjust the hardware clock $H^i(t)$ by itself but the logical clock can be corrected. It is calculated as follows:

$$L^i(t) = \int_{t_0}^{t} h^i(\tau)l^i(\tau)d\tau + \Psi^i(t_0), \qquad (3)$$

where $L^i(t)$ is the logical clock of Node i. $l^i(t)$ is the relative logical clock rate. $\Psi^i(t_0)$ is the offset between the logical clock and the hardware clock on Node i at time t_0.

Besides drift, short-term instability of crystal oscillator is more nondeterministic. According to [18], as illustrated in Fig. 2, drift is a kind of effect that is observed over long periods of time (hours, days or years), whereas the other is random and observed over periods that are typically measured in fractions of a second to minutes.

Discussions about $h(t)$

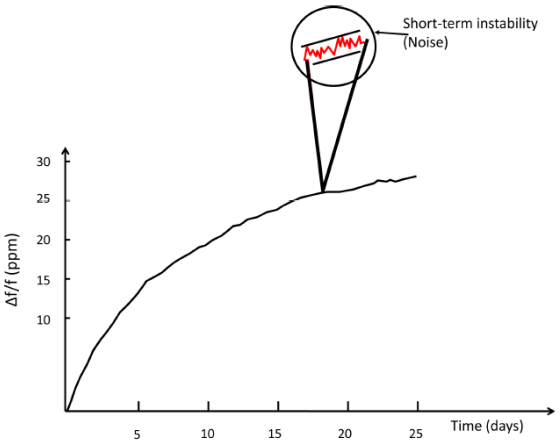

Fig. 2. Aging and short-term stability [18].

The instantaneous frequency $h(t)$ of the practical crystal oscillator with noise can be expressed as [19],

$$h(t) = h_c(t) + \frac{1}{2\pi}\frac{d\phi(t)}{dt}, \left|\frac{\phi(t)}{2\pi h_c(t)}\right| \ll 1, \quad (4)$$

where the current frequency of crystal oscillator $h_c(t)$ can be regarded as constant h_c since it drifts with time slowly. The deviation of phase from the ideal is $\phi(t)$. Generally, $\phi(t)$ consists of white and flicker noises taking the form of phase jitters which may reflect jitters of message. The goal of time synchronization is to share identical time scale through achieving the same clock skews $h(t) \cdot l(t)$ and proper offsets $\Psi^i(t_0)$. The clock skews are designed to compensate the drift rather than stochastic short-term instability. Conversely, skews estimation can be worsened by both jitters and clock granularity. The latter is stated in *Clock Granularity* of 3.2.

Nodes in Network

Time synchronization in large-scale WSN is one of the many challenges. Multihops between nodes and the root node (reference) become another factor degrading the precision of skews estimation. Thus multihops scenario is considered. For simplicity, given three nodes $i-1$, i and $i+1$ in link-like topology, Node i received messages from the predecessor Node $i-1$ and synchronize to it. The similar processes occur between Node i and its successor $i+1$. The distance between two nodes is defined as the hops. For instance, the distance that is denoted as d between Node 2 and Node 5 is three.

The ultimate goal of time synchronization in one network with n nodes is that all the logical clocks and all the clock skews $h(t) \cdot l(t)$ are both identical at time t.

$$\begin{cases} L^1(t) = L^2(t) = \cdots = L^n(t) \\ h^1(t) \cdot l^1(t) = h^2(t) \cdot l^2(t) = \cdots = h^n(t) \cdot l^n(t) \end{cases} \quad (5)$$

Generally, there is only one root node r in the network to which all the others synchronize themselves. Thereupon, Equations (5) can be re-written more precisely as follows:

$$\begin{cases} L^1(t) = L^2(t) = \cdots = L^n(t) = H^r(t) \\ h^1(t) \cdot l^{1\to r}(t) \\ = h^2(t) \cdot l^{2\to r}(t) = \cdots = h^n(t) \cdot l^{n\to r}(t) = h^r(t) \end{cases}, \quad (6)$$

where for Node n, $l^{n\to r}(t)$ is the relative logical clock rate to the root Node r.

Multihops error that is induced by clock granularity will be discussed below.

3.2. Multihops Error

First of all, some conclusions in [6] will be described here.

3.2.1. About PulseSync

According to [6], in the case of $d = 1$, the probability that relative logical clock rate $\hat{l}^{i\to r}(t)$ computed by Node i has an error of at least $\Omega(j/Bk^{3/2})$ is constant:

$$P\left[\left|\hat{l}^{i\to r}(t) - l^{i\to r}(t)\right| \geq \Omega\left(\frac{j}{Bk^{3/2}}\right)\right] \geq \frac{1}{2}, \quad (7)$$

where B is the PulseSync's synchronization periods. The jitter of messages meets the Gaussian distribution[2] with a standard deviation J or uniformly distributed in $[-j, j]$. Each node estimates the logical relative rate based on k observation values. $\hat{l}^{i\to r}(t)$ is estimated by Node i and $l^{i\to r}(t)$ is the real value. For better description, $j/Bk^{3/2}$ is denoted as $E(j)$.

When $d > 1$, the error of relative logical clock rate $\hat{l}^{i\to r}(t)$ on Node i is,

$$E(j) = \frac{j\sqrt{d}}{Bk^{3/2}} \quad (8)$$

The lower bound (the maximum difference of logical clock value between any two nodes) of PulseSync is as follows:

$$G(t) = \Omega\left(\frac{j\sqrt{D}}{\sqrt{k}}\right), \quad (9)$$

where D is the diameter of the network N. The effect of clock granularity ignored in PulseSync [6] will be delved below.

3.2.2. Clock Granularity

In practice, the clock value of WSN node is discrete. When receiving two identical clock values, the node hardly makes a distinction between them and brings quantization error that will be amplified through multihops as with the message jitter. Multihops error is composed of message jitter and clock granularity.

Therefore, the jitter should be represented as follows:

$$\left\lceil\frac{T_{jitter}}{T_{gran}}\right\rceil \cdot T_{gran} + \upsilon \cdot T_{gran}, \quad (10)$$

where T_{gran} is clock granularity and suppose that υ is a Bernoulli distributed random variable with mean of

[2] Jitter is not exactly Gaussian Distribution, but we can make it zero mean and the resulting behavior is very similar.

0.5. T_{jitter} is the time of jitter. $\lceil\cdot\rceil$ is the symbol of rounding-up. Consequently, when considering the discreteness of t, $E(j)$ could be represented as follows:

$$E(j) = \frac{\left(\left\lceil \frac{j}{T_{gran}} \right\rceil + 0.5\right) \cdot T_{gran}}{Bk^{3/2}} \tag{11}$$

Similarly, like Equation (8), $E(j)$ is as follows on the condition of $d > 1$:

$$E(j) = \frac{\sqrt{d} \cdot \left(\left\lceil \frac{j}{T_{gran}} \right\rceil + 0.5\right) \cdot T_{gran}}{Bk^{3/2}}, \tag{12}$$

where $\frac{\left(\left\lceil \frac{j}{T_{gran}} \right\rceil + 0.5\right) \cdot T_{gran}}{Bk^{3/2}}$ is denoted as $\gamma(j)$.

As a result, in practice, the lower bound of PulseSync should be as follows:

$$G(t) = \Omega\left(\sqrt{D}\gamma(j)Bk\right) \tag{13}$$

Equation (11) shows two points:

a) $\gamma(j)$ will exists even though j is minor enough to be ignored;

b) If T_{gran} is small enough, $\gamma(j) \approx j$; on the other hand, if $T_{gran} \gg j$, the numerator of $\gamma(j)$ is governed by T_{gran}.

The first point indicates that the error of relative logical rate estimation will exists even though j tends to zero. The second implies that jitters j can be regarded as $\gamma(j)$ when the clock resolution is high enough. Conversely, clock granularity will take on a more dominant factor and can not be neglected any more as it is coarser. It is worth noting that, sometimes, the errors in real time (not in clock ticks) will significantly reduce the synchronization accuracy due to the coarse clock granularity. Equation (12) means that the multihops error cannot be annihilated in PulseSync as long as t is discrete. The clock granularity T_{gran} dominates the synchronization error in that multihops synchronization error exists even if the jitter tends to zero.

Equation (11) suggests that $E(j)$ is proportional to T_{gran}. However, generally speaking, reducing clock granularity indicates increasing operating frequency and leading to significant amount of node's power consumption as a result. In contrast, this paper argue that the clock granularity impact could be relieved by filtering and delivering the relative logical clock rate without changing T_{gran}.

4. Improved PulseSync

The improvements are presented in this section. We describe the main idea of the algorithm in 4.1 and do some explanations concerning the filter in 4.2.

4.1. Algorithm Description

In general, for one node, the process of synchronization needs to estimate the relative logical clock rate and the offset between itself and the root node that is the reference. In the algorithm, the offset value $\Psi^i(t_0)$ of Node i in Equation (3) is obtained in the same way as PulseSync. With the purpose of abating the impact of the clock granularity, the work is mainly devoted to getting the relative logical rate improved throughout four steps. Nodes are arranged as link topology and messages flow as Fig. 3.

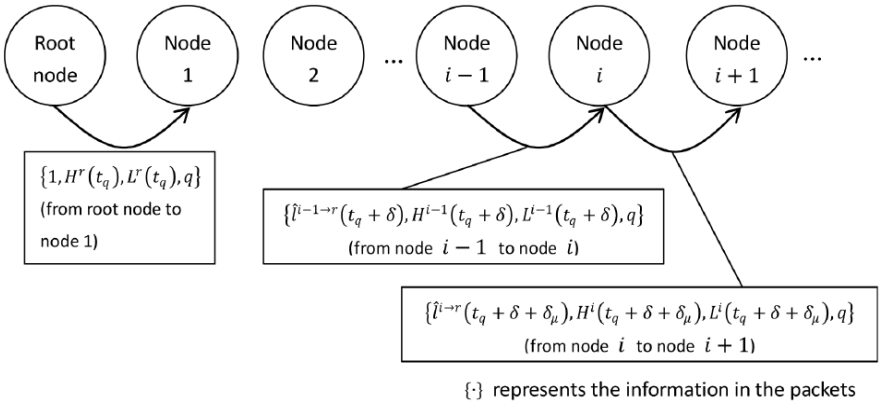

Fig. 3. Synchronization message flow diagram.

Each node regards the root node as the only reference. $L^r(t_q)$ ought to become the global clock as all the nodes in the network synchronizing successfully. Message $\{1, H^r(t_q), L^r(t_q), q\}$ with the sequence number q is broadcasted by the root node at time t_q. Then it is received by Node i at time $t_q + \delta$. δ_μ is very small since Node i delivers it promptly at time $t_q + \delta + \delta_\mu$ once receiving the qth message.

The main procedures of the algorithm will be stated below.

***Step 1**: Node i refreshes $\hat{l}^{i\to i-1}(t_q+\delta)$ and $\Psi^i(t_q+\delta)$ once receiving message.*

When receiving the synchronization message from Node $i-1$, Node i calculates $\hat{l}^{i\to i-1}(t_q+\delta)$ using $H^{i-1}(t_q+\delta)$ contained in the packet and $\Psi^i(t_q+\delta)$ through $L^{i-1}(t_q+\delta)$. Therefore,

$$h^{i-1}(t_q+\delta) = h^i(t_q+\delta)\cdot\hat{l}^{i\to i-1}(t_q+\delta) \tag{14}$$

***Step 2**: Node i smooths $\hat{l}^{i\to i-1}$.*

Node i gets $\bar{l}^{i\to i-1}(t_q+\delta)$ as the result of smoothing $\hat{l}^{i\to i-1}(t_q+\delta)$ with Kalman Filter.

***Step 3**: Node i calculates $\hat{l}^{i\to r}(t_q+\delta)$.*

$\hat{l}^{i\to r}(t_q+\delta)$ can be get as the following:

$$\hat{l}^{i\to r} = \hat{l}^{i-1\to r}\cdot\bar{l}^{i\to i-1}, \tag{15}$$

where $\bar{l}^{i\to i-1}\cdot h^i = h^{i-1}$. If the distance between Node i and root Node r is d, $\hat{l}^{i\to r}$ is boiled down to product of all relative logical clock rates on the routine from r to i, i.e.,

$$\hat{l}^{i\to r} = \left(\prod_{n=2}^{d}\bar{l}^{n\to n-1}\right)\cdot\left(\hat{l}^{1\to r}\right) \tag{16}$$

***Step 4**: Node i encapsulates $\{\hat{l}^{i\to r}(t_q+\delta+\delta_\mu), H^i(t_q+\delta+\delta_\mu), L^i(t_q+\delta+\delta_\mu), q\}$ and delivers it.*

It is noted that $\hat{l}^{i\to r}(t_q+\delta+\delta_\mu)$ is equivalent to $\hat{l}^{i\to r}(t_q+\delta)$ since the time δ_μ is quite short. According to the Equation (3), $L^i(t)$ can be written as follows:

$$L^i(t) = \int_{t_q+\delta}^{t} h^i(\tau)\cdot\hat{l}^{i\to r}(t_q+\delta)\,d\tau + \Psi^i(t_q+\delta), \quad t > t_q+\delta \tag{17}$$

The pseudo-code is as follows (Fig. 4). The sequence number q is to avoid duplicated message flooding in networks. On the root node, only broadcasting reference messages and refreshing the sequence number q are required. Ordinary nodes check the sequence number q to ensure that the message is new and maintain the *Queue* (a circular queue with k-length) in the receiving interrupt service routine. Function *refresh* is called to calculate *psi* and $\hat{l}^{i\to i-1}$ and accomplish sending when the *receivingInterrupt* exits. Function *RTimer_NOW* returns the hardware clock. In short, filtering improves the single-hop synchronization performance and delivering relative logical clock rates makes the filtering effect propagated through multihops.

For the root node

```
If(synchTimeIsExpired)
{
 send(1, Hʳ, Lʳ, q);
 q++;
}
```

For ordinary nodes

```
receivingInterrupt()
{
 ...
  if(newIncoming(l̂^{i-1→r}, H^{i-1}, L^{i-1}, q) AND q > q')
  {
    q' = q;
    addOneItemIntoQueue(H^i, H^{i-1}, L^{i-1}, q, Queue);
    removeOneItemFromQueue(H^i, H^{i-1}, L^{i-1}, q - k, Queue);
    pollFunctionAfterInterrupt(*refresh);
  }
  ...
}
refresh(Queue)
{
  (l̂^{i→i-1}, psi)=calculate(Queue);
  l̂^{i→r} = l̂^{i-1→r} * kalmanFilter(l̂^{i→i-1});
  /*
    H^i = RTimer_NOW();
    L^i = l̂^{i→r} * H^i + psi;
  */
  send(l̂^{i→r}, H^i, L^i, q);
}
```

Fig. 4. Pseudo-code of improved PulseSync.

In order to explain the difference between PulseSync and the improved one, the comparisons are illustrated as Fig. 5.

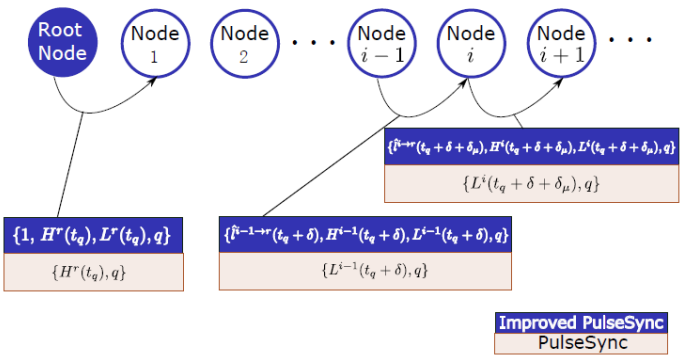

Fig. 5. PulseSync and the improved PulseSync.

Two critical differences can be pointed:

1) The relative logical clock rate besides the clock value is delivered straightly to mitigate the deviation of the estimated clock skew;

2) A filter is imposed on relative logical clock rate to abate the impact of discreteness of clock value.

4.2. The Reason of Utilizing Filter

As discussed in 3.1, the target of skews estimation in time synchronization is to balance drifts, which is a relatively gradual and continuous process, instead of the short-term noises (jitters). For Node i,

$$\frac{dh^i(t)}{dt} \in [0-\varsigma, 0+\varsigma], \tag{18}$$

where $\varsigma \to 0$ and $\varsigma \neq 0$. As Equation (14), the relation between $l^{i+1\to i}$ and h^i is

$$h^i = h^{i+1} \cdot l^{i+1\to i} \tag{19}$$

If we differentiate Equation (19) on both sides,

$$\frac{dh^i}{dt} = \frac{dh^{i+1}}{dt} l^{i+1\to i} + \frac{dl^{i+1\to i}}{dt} h^{i+1}, \tag{20}$$

where both $l^{i+1\to i}$ and h^{i+1} tend to one. Therefore, combined Equation (20) with (18), the change rate of relative logical clock rate $l^{i+1\to i}$ is

$$\frac{dl^{i+1\to i}(t)}{dt} \in [0-2\varsigma, 0+2\varsigma] \tag{21}$$

From Equation (21), $l^{i+1\to i}$ is comparatively stable in very short time, motivating us to smooth out the noise by statistics filtering. For instance, Kalman filter is feasible. Detail model and parameters will be expounded in Section 5.4.

So far, the improved algorithm has been proposed and the next section will focus on implementation.

5. Implementation and Simulation

In this section, the simulation platform and implementation are delineated.

5.1. Simulation Platform

The simulation platform adopted in this paper is Cooja Simulator [20]. A simulated Contiki Mote in COOJA executes Contiki [21] operating system. Generally, the code for specific platform could be easily ported to Motes in Cooja without modifying for developers. EXP5438 platform is chosen as the mote in Cooja. The platform EXP5438 is equipped with MSP430F5438 and RF chip CC2420. The former is an ultra-low power MCU with flash 256 Kb and RAM 16 Kb from Texas Instruments. The latter is a packet-oriented RF chip and the modulation mode is configured to FSK. The system frequency of the EXP5438 is set to 8 MHz by multiplying the signal from a 32.768 kHz external quartz crystal using a phased locked loop. Also, the crystal oscillator 32.768 kHz is used as the clock to be synchronized. Moreover, in Cooja, the operating frequency of all nodes is 7995492 Hz actually and there exists jitter of messages on every mote but no clock drift. It is due to zero clock drift and same frequency of all the nodes that the skew $l^{i\to r} \cdot h^i$ is completely dominated by the estimated relative logical clock rate $\hat{l}^{i\to r}$.

5.2. Implementation in Contiki

According to Contiki, the time synchronization is schemed as Fig. 6. For reducing the communication overhead, the head of time synchronization packet should be as short as possible. Hence, the communication of time synchronization is completed in MAC Layer, which brings synchronization independent of the protocols (like TCP/IP) in upper layers.

In Contiki, exactly speaking, the timestamping operation lying on the driver of RF eliminates the stochastic MAC access time. In the CSMA mechanism, when node sending packet, the node will perform the Clear Channel Assessment (CCA). If CCA is valid and the packet with synchronization information is ready to transmit (SFD pin is 1), the node will timestamp in the packet. To avoid failure of writing timestamps in time, some paddings are added to the head. When receiving packet, node will not call the corresponding ISR until the SFD is 1. Timestamping executes in the ISR.

The synchronized logical clock will be provided to applications in the form of API.

The hardware clock is 32.768 kHz. It shows that the clock granularity T_{gran} is 1/32768 seconds. And the waiting intervals before sending in both PulseSync and the improved are set to zero.

5.3. Simulation

Since the algorithm in this paper is aimed to improve the multihops error in time synchronization, chain topology (Fig. 7) is used to minimize the other interferences. Each node can only communicate with adjacent node in the chain topology (Fig. 7). Node 1 is the root node which is the time reference

Here, a testing node broadcasting packets in periods *B* is devised to facilitate evaluating the time synchronization result. The requirement of testing node is capable of covering all the nodes in the experiments while denying replies to any common node in the network.

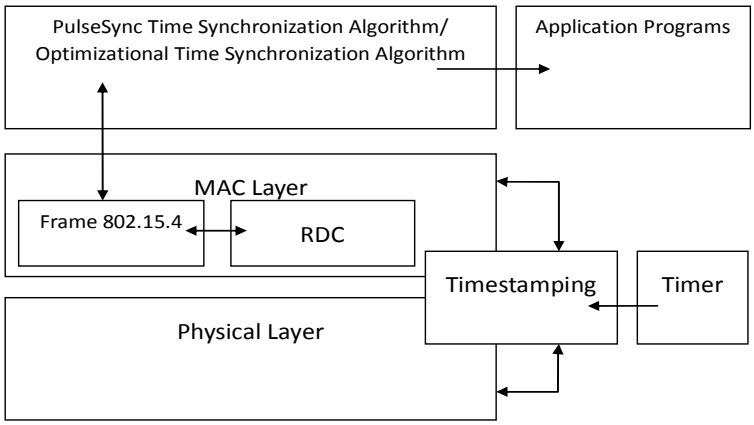

Fig. 6. The schematic diagram in Contiki.

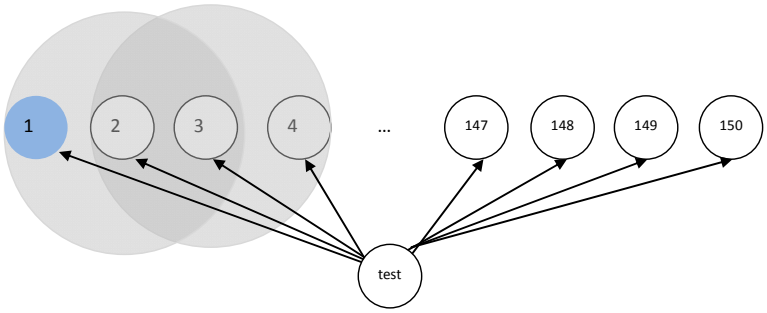

Fig. 7. The schematic diagram in Contiki.

To cater for the demands and cut down work loads, the testing node holds larger wireless coverage than other common nodes and disables receiving. Upon receiving the testing packet, the others will timestamp and print the related information (including estimated current relative logical clock rate to the root node, the estimated current logical clock, etc.) to the console of Cooja.

Moreover, 150 nodes (exclude testing node) are deployed in totally and the network diameter D is 149. Both the period of PulseSync and that of our algorithm are all 20 seconds. The Radio Duty Cycling (RDC) of all nodes is configured to ContikiMAC. Broadcasting from testing node is scheduled between two synchronization actions. The testing broadcast period is also 20 seconds. To agree with practical situations, the *rpl-collect* application program is loaded and run. The total length of simulation is 3 hours. The main setups are collected in Table 1.

5.4. About the Kalman Filter

Combined with Equation (21), the process model of the Kalman Filter can be represented as:

$$\begin{cases} l^{i\to i-1}(t_m) = l^{i\to i-1}(t_{m-1}) + \omega_{m-1} \\ l_{obs}^{i\to i-1}(t_m) = l^{i\to i-1}(t_m) + \varphi_m \end{cases}, \qquad (22)$$

where $l^{i\to i-1}(t_m)$ is the system state which denotes the relative logical clock rate on node i at time t_m. $l_{obs}^{i\to i-1}(t_m)$ is the observation of the relative logical clock rate on Node i at time t_m.

Table 1. Simulation Setups.

Network	
Nodes number	150
Topology	link
Running time	180 mins
Nodes	
Quartz crystal oscillator frequency of nodes	32.786 kHz
System frequency of nodes	8 MHz(using PLL from the crystal, 7995492 Hz actually)
Operating System on nodes	Contiki 2.6
MAC Layer(RDC)	ContikiMAC(RDC)
MAC Layer	CSMA
Applications running while simulating	Rpl-collect
Synchronization	
Beacon periods B	20 s
Observation value number k	8
Hardware clock rate	32.768 kHz (no drifts but jitters exist in this simulation)
Clock granularity	1/32768 s
Waiting time before sending	0
Smooth method	Kalman filter $(2\varsigma = 10^{-9}, \gamma(j) = 10^{-7})$

The interval between t_m and t_{m-1} stands for the synchronization periods B. φ_m is the noise produced by discreteness of clock value, i.e.,

$$\varphi_m \sim N(0, \gamma(j)) \tag{23}$$

ω_{m-1} is the process noise which is Gaussian distributed with standard deviation 2ς, i.e.,

$$\omega_m \sim N(0, 2\varsigma) \tag{24}$$

In this simulation, 2ς is assumed to 10^{-9} due to the crystal drift is gradual and continuous when only affected by environment. Moreover, taking the RAM and communication ability into account, let $k = 8$, B= 20 s. In Fig. 7, the standard deviation of estimated skew on Node 2 with PulseSync (red curve) is 9.5302×10^{-8}. Thus, $\gamma(j)$ can be set to 10^{-7}.

5.5. Simulation of the Kalman Filter

Finally, the parameters of the Kalman filter are set as follows:

$$\begin{cases} 2\varsigma = 10^{-9} \\ \gamma(j) = 10^{-7} \end{cases} \tag{25}$$

The skews $l^{2\to r} \cdot h^2|_{h^2=1}$ that the Node 2 computes with PulseSync and the improved algorithm are plotted on the Fig. 8. The blue curve is h^2 ($h^2 = 1$). Apparently, the filter is very effective and the standard deviation of the skew smoothed by the Kalman Filter only is limited to 2.137×10^{-9}.

5.6. Results

First, PulseSync is employed to explore the relationships between average time error and average estimated skew error, illustrated in Fig. 9.

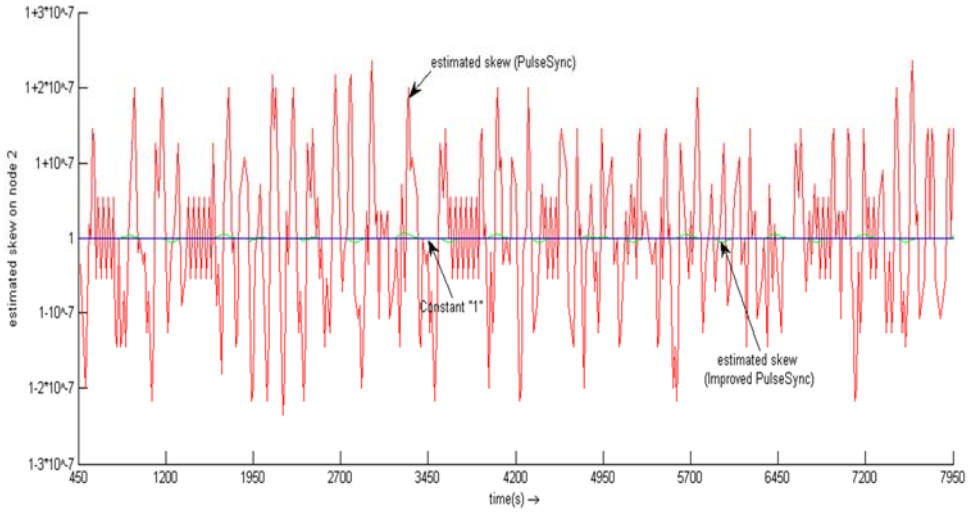

Fig. 8. The skew computed on Node 2: PulseSync (red) vs. Improved PulseSync (green).

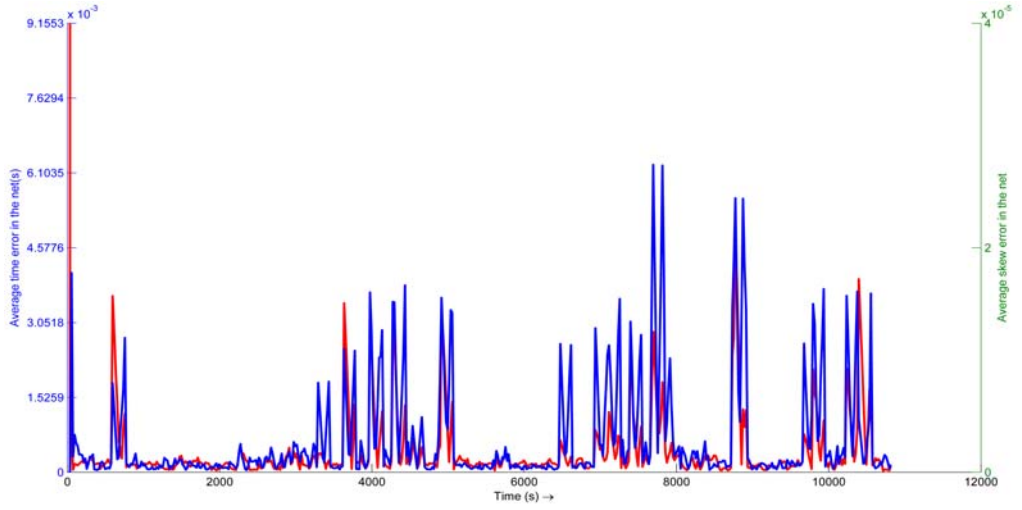

Fig. 9. PulseSync: Average time error in net (red) vs. average skew error in net (blue).

The average time error in net is defined as follows:

$$T_{ATE} = \frac{\sum_{i\neq r, i\in N}\left|L^i(t) - L^r(t)\right|}{Num-1}, \quad (26)$$

where Num is the number of nodes in the network. In this simulation, $L^r(t)$ is $L^1(t)$ actually since the Node 1 is the root node. The average skew error in net is defined as follows:

$$Skew_{ATE} = \frac{\sum_{i\neq r, i\in N}\left|\hat{l}^{i\to r}(t) - 1\right|}{Num - 1} \quad (27)$$

The global average time error varies with the average estimated skew error as shown in Fig. 8, especially at time 596, 3636, 7696, 8776 and 10400 seconds. On the other words, the more precise estimated skew, the less synchronization time error. The improved one has an evident superiority in skew error as illustrated in Fig. 11 and slight better performance in time error in Fig. 10 since that the effect of skew error has not been amplified through the hops yet.

The comparisons of average time error in net and neighbor nodes between PulseSync and ours are shown in Fig. 12, Fig. 13 respectively. The average time error between neighbors in net is defined as follows:

$$T_{ATNE} = \frac{\sum_{i\neq r, i\in N}\left|L^i(\mathrm{t}) - L^{i-1}(\mathrm{t})\right|}{Num - 1} \quad (28)$$

It is clearly observed that, with the improved PulseSync, the time error (in Fig. 12) does not fluctuate so much and performs well in view of time. The means of the error during the three hours are 8.8886×10^{-4}s with PulseSync and 7.6954×10^{-4}s with the improved one respectively. And the standard deviations of the error during the test periods, which are used to show how stable the synchronized network are, are 6.5869×10^{-4}s (red curve) and 3.0191×10^{-4}s (green curve).

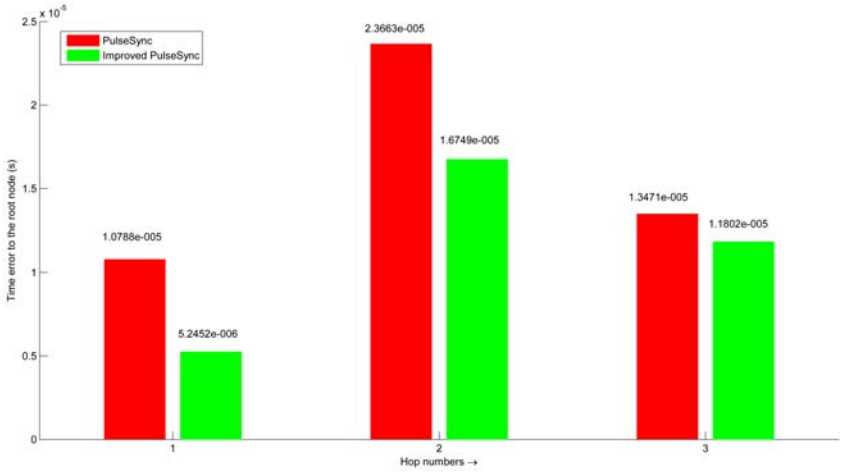

Fig. 10. The time error to the root node with hop numbers: PulseSync (red) vs. Improved PulseSync (green): the first three hops.

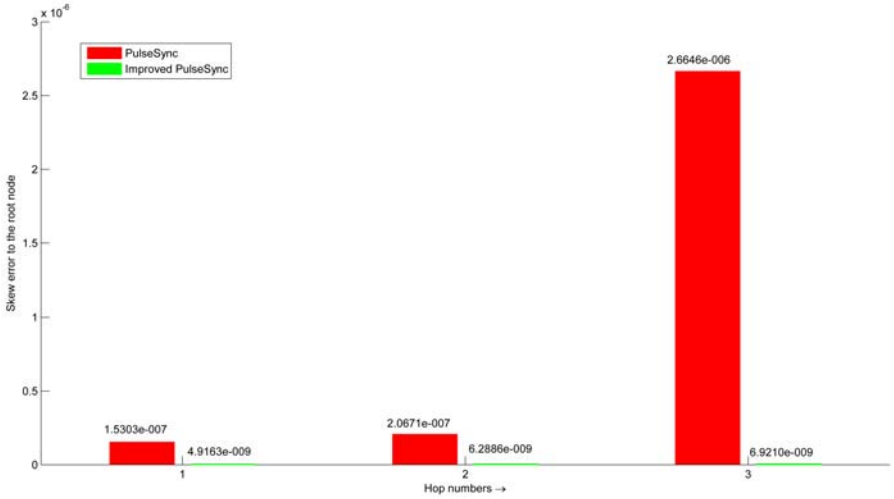

Fig. 11. The skew error to the root node with hop numbers: PulseSync (red) vs. Improved PulseSync (green): the first three hops.

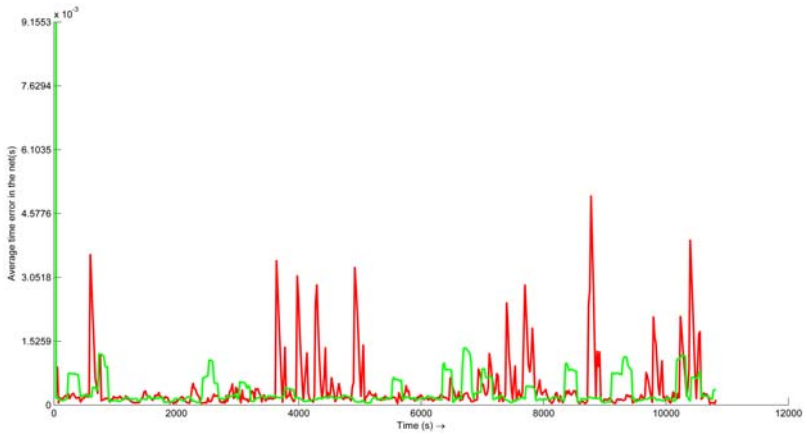

Fig. 12. The average time error in net: PulseSync (red) vs. Improved PulseSync (green).

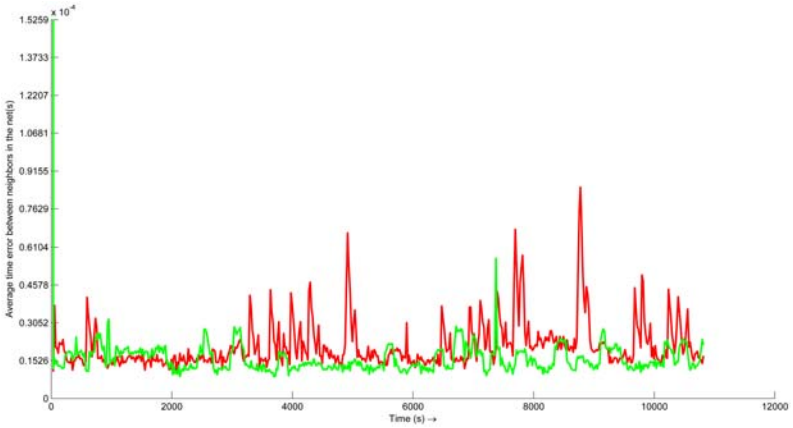

Fig. 13. The average time error between neighbors in net: PulseSync (red) vs. Improved PulseSync (green).

T_{ATNE} shows the synchronization quality among neighbor nodes. The factor that Kalman filter stabilizes the relative logical clock rate can be observed in Fig. 8 and the effects on the time plotted in Fig. 13 are tangible as well. The mean and standard deviation of the error during the test periods are 6.6326×10^{-4}s and 9.5423×10^{-6}s with PulseSync, 6.5798×10^{-4}s and 4.8146×10^{-6}s with the improved PulseSync.

Similarly, from the view of hops, the time error and skew error deviation over multihops decline remarkably with our algorithm, which can be figured out from Fig. 14, Fig. 15. The difference in skew error is evident at the very start (after 2 hops), also in time error, the slopes of the zero-crossing linear fitting are 3.9940×10^{-6} and 5.6939×10^{-6} respectively. The improved algorithm minimizes the growth rate of time error with the number of hops. Essentially, these are led by alleviating the clock granularity.

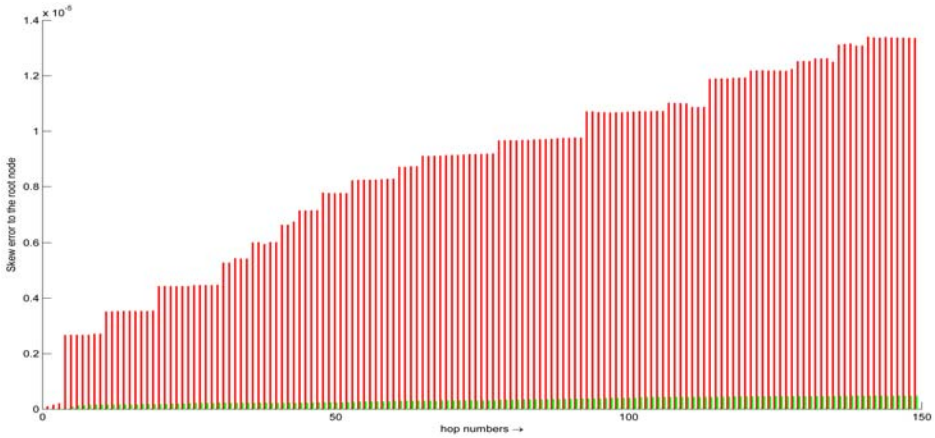

Fig. 14. The skew error to the root node with hop numbers: PulseSync (red) vs. Improved PulseSync (green).

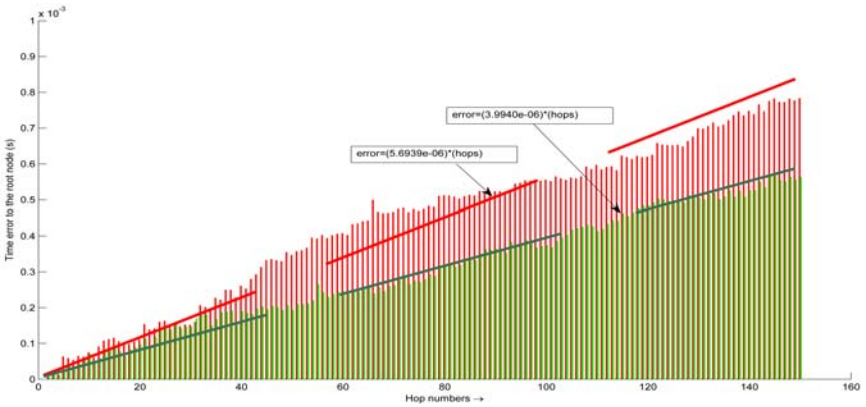

Fig. 15. The time error to the root node with hop numbers: PulseSync (red) vs. Improved PulseSync (green).

6. Conclusions

In this work, an improved algorithm based on PulseSync is proposed. Clock granularity impact is alleviated by filtering and delivering relative logical clock rate, which is beneficial to reducing the deviation of estimated clock skew through several hops and avails the accuracy of time synchronization ultimately. The algorithm is implemented in Contiki OS and simulated in Cooja. Theoretical and experimental analyses show that the novelty on PulseSync works well in minimizing the multihops error. In the future, we may revise more sophisticated filter models. Another direction is to find some better energy efficient synchronization communication mechanism.

Acknowledgements

The authors acknowledge the support provided by the Safety and Emergency Lab of Shanghai Advanced Research Institute (SARI) for this study. And this project is supported by ”the Next Generation of Information Technology (IT) for Sensing China” of Chinese Academy of Sciences (XDA06010800) and NSFC (61201446).

References

[1]. Somov A., *et al*., Development of wireless sensor network for combustible gas monitoring, *Sensors and Actuators A: Physical*, Vol. 171, No. 2, 2011, pp. 398-405.

[2]. Xiaoyuan M., W. Jianming, Towards Gas Density Measurement: Design and Implementation of the Application-Oriented WSN, *International Journal of Computer Science Issues*, Vol. 11, No. 5, 2014, pp. 1-149.

[3]. Martinez K., *et al*., Environmental sensor networks, *IEEE Computer*, Vol. 37, No. 8, 2004, pp. 50-56.

[4]. Cionca V., T. Newe, V. Dadârlat, TDMA protocol requirements for wireless sensor networks, in *Proceedings of the 2nd IEEE International Conference on Sensor Technologies and Applications (SENSORCOMM'08)*, 2008, pp. 30-35.

[5]. Mills D., *et al*., RFC5905: Network time protocol version 4: Protocol and algorithms specification, *Internet Engineering Task Force (IETF)*, June 2010, pp. 1-110.

[6]. Lenzen C., P. Sommer, R. Wattenhofer, Optimal clock synchronization in networks, in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, 2009, pp. 235-238.

[7]. Lenzen C., P. Sommer, R. Wattenhofer, PulseSync: An efficient and scalable clock synchronization protocol, *IEEE/ACM Transactions on Networking (TON)*, Vol. 23, No. 3, 2015, pp. 717-727.

[8]. Schmid T., P. Dutta, M. B. Srivastava, High-resolution, low-power time synchronization an oxymoron no more, in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2010.

[9]. Ganeriwal S., R. Kumar, M. B. Srivastava, Timing-sync protocol for sensor networks, in *Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems*, 2003, pp. 138-149.

[10]. Sichitiu M. L., C. Veerarittiphan, Simple, accurate time synchronization for wireless sensor networks, in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC' 03)*, Vol. 2, pp. 1266 - 1273.

[11]. Elson J., L. Girod, D. Estrin, Fine-grained network time synchronization using reference broadcasts, *ACM SIGOPS Operating Systems Review*, Vol. 36, Issue SI, 2002, pp. 147-163.

[12]. Maróti M., *et al.*, The flooding time synchronization protocol, in *Proceedings of the 2nd ACM International Conference on Embedded Networked Sensor Systems*, 2004.

[13]. Fan R., N. Lynch, Gradient clock synchronization, *Distributed Computing*, Vol. 18, No. 4, 2006, pp. 255-266.

[14]. Sommer P., R. Wattenhofer, Gradient clock synchronization in wireless sensor networks, in *Proceedings of the IEEE International Conference on Information Processing in Sensor Networks*, 2009, pp. 37-48.

[15]. Du J., Y.-C. Wu, Distributed clock skew and offset estimation in wireless sensor networks: asynchronous algorithm and convergence analysis, *IEEE Transactions on Wireless Communications*, Vol. 12, No. 11, 2013, pp. 5908-5917.

[16]. Luo B., Y. C. Wu, Distributed clock parameters tracking in wireless sensor network, *IEEE*

Transactions on Wireless Communications, Vol. 12, No. 12, 2013, pp. 6464-6475.
[17]. Wu Y.-C., Q. Chaudhari, E. Serpedin, Clock synchronization of wireless sensor networks, *IEEE Signal Processing Magazine*, Vol. 28, No. 1, 2011, pp. 124-138.
[18]. Vig J. R., Quartz crystal resonators and oscillators, *US Army Communications-Electronics Research, Development & Engineering Center Fort Monmouth*, NJ, USA, 2004.
[19]. Stein S. R., Frequency and Time-Their Measurement and Characterization, *Precision Frequency Control*, Vol. 2, Chap. 12, 1985, pp. 191-232.
[20]. Eriksson J., *et al*., COOJA/MSPSim: interoperability testing for wireless sensor networks, in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (ICST'09)*, 2009.
[21]. Dunkels A., B. Grönvall, T. Voigt, Contiki-a lightweight and flexible operating system for tiny networked sensors, in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, 2004, pp. 455-462.
[22]. Xiaoyuan Ma, Weisheng Tang, Jianming Wei, Jun Huang, Bo Zhang, Beyond PulseSync: Less Clock Granularity Impact, More Synchronization Accuracy, in *Proceedings of the 5th International Confererence on Sensor Networks (SENSORNETS 2016)*, Rome, Italy, 2016.