Sensors **2014**, *14*, 14744-14764; doi:10.3390/s140814744

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.com/journal/sensors

Article

A Group Neighborhood Average Clock Synchronization Protocol for Wireless Sensor Networks

Lin Lin [1],*, Shiwei Ma [1] and Maode Ma [2]

[1] Department of Automation, Shanghai University, Shanghai 200072, China; E-Mail: masw@shu.edu.cn

[2] School of Electrical and Electronic Engineering, Nanyang Technological University, 639798 Singapore; E-Mail: EMDMA@ntu.edu.sg

* Author to whom correspondence should be addressed; E-Mail: fxlinlin@shu.edu.cn; Tel.: +86-21-5633-1278; Fax: +86-21-5633-1183.

Received: 12 May 2014; in revised form: 21 July 2014 / Accepted: 4 August 2014 / Published: 12 August 2014

Abstract: Clock synchronization is a very important issue for the applications of wireless sensor networks. The sensors need to keep a strict clock so that users can know exactly what happens in the monitoring area at the same time. This paper proposes a novel internal distributed clock synchronization solution using group neighborhood average. Each sensor node collects the offset and skew rate of the neighbors. Group averaging of offset and skew rate value are calculated instead of conventional point-to-point averaging method. The sensor node then returns compensated value back to the neighbors. The propagation delay is considered and compensated. The analytical analysis of offset and skew compensation is presented. Simulation results validate the effectiveness of the protocol and reveal that the protocol allows sensor networks to quickly establish a consensus clock and maintain a small deviation from the consensus clock.

Keywords: wireless sensor networks; clock synchronization; convergence; consensus clock

1. Introduction

Wireless sensor networks (WSNs) are now widely deployed in homes, hospitals, factories, streets, battlefields, *etc.*, where they are used for a lot of monitoring applications, such as monitoring of water distribution systems [1], earthquakes [2], buildings [3], civil infrastructure [4], habitat activities [5], *etc.*

In some of the applications, such as data fusion, target tracking and speed estimation, the system needs to have the knowledge of time among sensor nodes in order to determine the timing of the event. This is realized by clock synchronization (sometimes called time synchronization). Besides the application requirement, clock synchronization is important to avoid interference if the medium access control protocol uses time division medium access (TDMA) [6,7]. For the networks that utilize duty-cycling schemes and turn off the radio for energy saving, accurate time helps to save energy by shortening the necessary guard time.

Imperfections in low-cost sensor nodes will cause offsets and skew drifts. Environmental factors such as the temperature also contribute to this clock shift. Therefore, the sensors will lose synchronization with other sensor nodes. This leads to the meaninglessness of sensing data for many applications. Clock synchronization techniques are therefore needed to synchronize the sensor nodes in WSNs. The algorithms compensate the offset and skew rate by exchanging information among sensor nodes. There are several challenges to design a clock synchronization algorithm. First, since the information is shared among sensor nodes by passage exchanging, propagation delays would cause inaccuracy of time instant sharing, so a specific compensation technique is required. Second, because the sensor nodes are densely deployed in the monitoring area, the nodes can only communicate with the sink node via a multi-hop route. Designing an accurate, fast convergence and scalable global clock synchronization algorithm becomes a challenge. Third, the sensors are inexpensive devices, which may often incur failure. A dynamic and robust scheme is needed. Fourth, since sensor nodes are almost always powered by batteries, energy efficiency becomes one of the major concerns. These challenges need to be addressed in the design of any clock synchronization protocol.

In this paper, a novel distributed clock synchronization scheme, named group neighborhood averaging (GNA), is presented. It is a fully distributed, network-wide clock synchronization algorithm. Two-way communication is adopted to estimate the propagation delay and further used for offset and skew rate compensation. Without extra data exchange, just using the exchanged messages, the algorithm takes average of the offset and skew rate value among the neighboring nodes within the transmission range. The group-averaging algorithm leads to a fast convergence.

The rest of this paper is organized as follows: Section 2 discusses related work. Section 3 presents the system model and Section 4 explains the proposed GNA algorithm. Simulation results are given and discussed in Section 5. Section 6 concludes the paper and describes possible future work.

2. Related Work

Many dedicated strategies and protocols have been proposed to address the problem of clock synchronization in WSNs in the past decades as surveyed in [8–11]. The Network Time Protocol (NTP) [12] is one of the popular clock synchronization protocols in use. It adopts a hierarchical tree of timeservers. By recording the sending and receiving timestamps, the propagation delay is estimated. The time is then compensated using this value. NTP is not suitable for WSNs for two main reasons. First, it takes long time for the entire network to synchronize to the reference clock. Second, the accuracy is low and the complexity is relatively high. Reference Broadcast Synchronization (RBS) [13] uses a node to send a reference broadcast beacon to neighbors in the physical layer. The receivers exchange the time of arrival and compensate their own clocks. The advantage is that RBS shortens the critical path since the broadcast beacon is only used as a reference. The potential drawback lies that the exchanges of time of arrival between the neighbors incur a large overhead and the performance may decrease in large-scale multi-hop networks. In [14], by recording the sending and receiving time of the probe message, the bounds of the relative offset and drift can be obtained and then the relative offset and relative drift are estimated. It is simply a computational task. The algorithm is based on a tree topology network and the sensor nodes only synchronize with their parent fusion nodes. The disadvantage of the protocol is that it is designed only for a tree topology network, not a mesh network which is more common. Another problem is that if synchronization were required for the entire network, the convergence time would be very large. Timing-sync protocol for sensor networks (TPSN) [15] and recursive time synchronization protocol (RTSP) [16] have similar solutions. The protocols periodically elect a root node, which acts as reference clock. Root node rotation balances the energy consumption among all the sensor nodes. A tree architecture is formed for clock synchronization. A similar NTP algorithm, which uses four time stamps to estimate the propagation delay, is adopted. The disadvantage is that the level discovery phase consumes quite a large amount of energy and time, which leads to slow convergence. In addition, due to the hierarchical topology, the protocol is prone to node failure. The flooding time synchronization protocol (FTSP) [17] also randomly elects a root node, but for time synchronization, either single hop or multi-hop, FTSP adopts one way synchronization. The message delivery delay is decomposed into interrupt handling of CPU, encoding and decoding of radio chip to transform the message, byte alignment of radio chip at the receiver side and the propagation delay. The time stamping effectively reduces the error introduced by interrupt handing, encoding and decoding. Compared to them, propagation is relatively small and neglected. Time-diffusion synchronization protocol (TDP) [18] uses NTP-like information exchange and spreads the reference time to the whole network. The performance is good, but the algorithm is complicated and not energy efficient. In summary, for tree hierarchical clock synchronizations, the disadvantages include slow convergence, energy waste and complexity of root node election, proneness to node and link failure and difficulties for mobile scenarios.

Distributed consensus-based protocols for global clock synchronization have been proposed in [19–23]. In [21], local synchronization is conducted by averaging the offset and finally the entire network converges to a virtual consensus clock, but only clock offset is compensated. Clock drift is not discussed. References [19] and [22] consider both offset and clock drift compensation. Each node periodically broadcasts a synchronization beacon. The receiving node takes an average of the offset and drift. The solution is fully distributed without the need of a root node. All nodes run the same algorithm. The protocol is robust to dynamic *ad-hoc* networks issues such as node and link failure. In [20], the algorithm is further improved. A confidence weight parameter is introduced for the average calculation. The node, which has conducted an averaging calculation, would be given a big weight. In [23], the protocol adopts max consensus to compensate for clock drift but the reason is not clearly presented. One limitation in the works presented in [19–22] is that the protocol is designed in the absence of propagation delay. However, the propagation delay is a very important issue, which cannot be ignored in real clock synchronization scenarios. Another limitation is that they take averaging only between one sender and one receiver at each time. This leads to a slow convergence. The authors of [24] use a group average and assume the propagation delay follows Gaussian distribution. However, the algorithm only achieves good performance when the network is time delay balanced. The reason is that the averaging operation for a Gaussian distributed delay approaches zero. This method would not be suitable for an unbalanced network. [25] considers both fixed and random propagation delays and proposes a distributed synchronization algorithm. By using the maximum likelihood estimation and belief propagation on graphical models, global clock synchronization is achieved. [26] improves it and proposes an asynchronous algorithm. Different nodes could update their estimates based on different frequencies. For these two methods, the algorithms are relatively complicated and need more storage and computational resources.

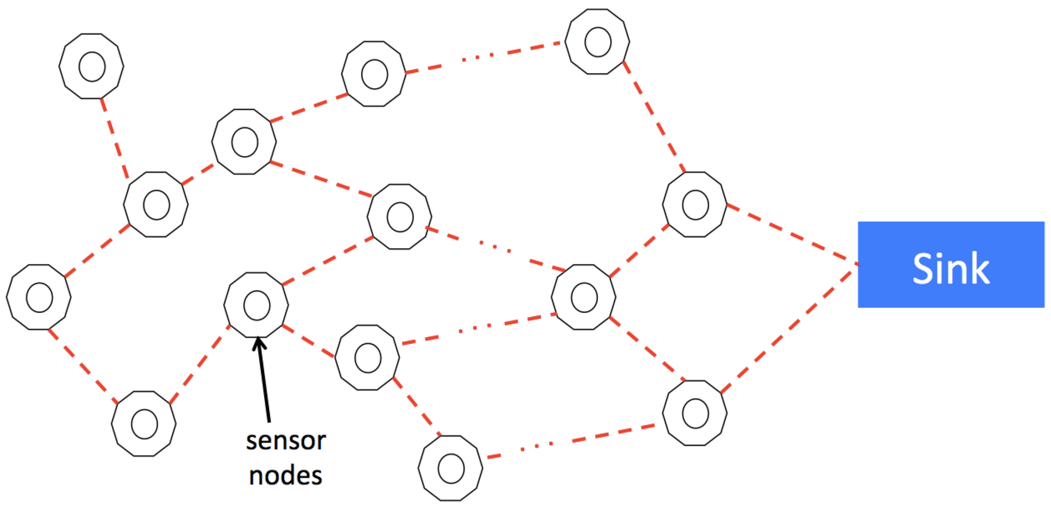
This paper proposes a fully distributed, network-wide clock synchronization scheme. The scheme is scalable and not constrained to a certain network topology. It is also robust to node failure. Two-way communication is adopted to estimate the propagation delay. The exchanged messages are multiply used for offset and skew rate compensation. Nodes take turns to ask their neighbors for clock information. Then they average all the received offset and skew rate values in each round and broadcast the values back to their neighbors. A fast convergence is achieved by the proposed group-averaging algorithm. The procedure of the algorithm is clearly presented. It has the advantages of accuracy, fast convergence, low complexity and easy implementation. The main contributions of this paper are:

(1) Group averaging is used instead of point-to-point averaging for faster convergence.
(2) Propagation delay is compensated compared with several important related papers.
(3) Minimum data exchange is achieved for both offset and skew rate compensation, and at the same time the propagation delay is considered and compensated.

3. System Model

A typical WSN is composed of tens, hundreds or even up to thousands of sensor nodes which are deployed in an *ad-hoc* fashion without careful planning, as shown in Figure 1. The sensor nodes collect and transmit the sensing data to a sink node. The sink node stores and processes the data. As discussed in Section 1, the sensor nodes need to store the time instants of the data collection so that later the sink node can aggregate and process the data from different sensor nodes. Therefore, the requirement that all the sensor nodes strictly keep a common clock becomes essential.

Figure 1. System model.



For each sensor node, we define the clock model as Equation (1). $C(t)$ is the clock reading at time *t*. α is the skew rate and β is the offset. In the ideal situation, α is equal to 1 and β is equal to 0:

$$C(t) = \alpha * t + \beta \tag{1}$$

The clocks in the sensor network may not be consistent for several reasons. First, the clock may drift, which can be caused by the manufacture of the oscillator components. Second, the clock could change due to some environmental condition such as temperature, pressure, battery voltage, *etc*. Third, the clocks from different sensor nodes may not be synchronized well at the beginning of the network deployment. All these reasons lead to clock readings varying from the real value. The clock reading $C_i(t)$ of node *i* is then expressed in Equation (2):

$$C_i(t) = \alpha_i * t + \beta_i \tag{2}$$

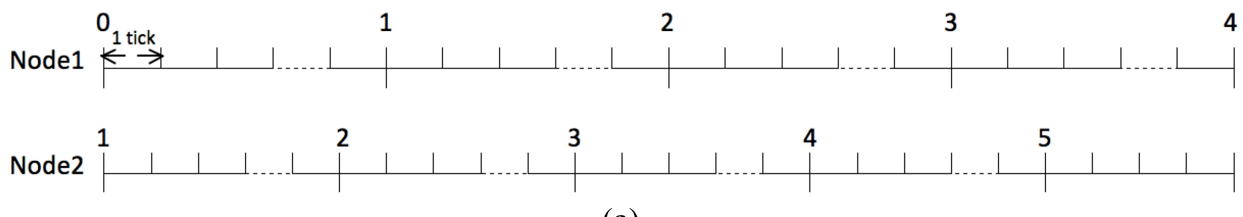
The aim is to find the compensated $\widehat{\alpha}_i$ and $\widehat{\beta}_i$, so as to make the compensated clock ($\hat{C}_i(t)$ expressed in Equation (3)) to converge to a consensus clock $C_c(t)$ (as shown in Equation (4)):

$$\hat{C}_i(t) = \frac{\alpha_i}{\widehat{\alpha}_i} * t + \beta_i - \widehat{\beta}_i \tag{3}$$

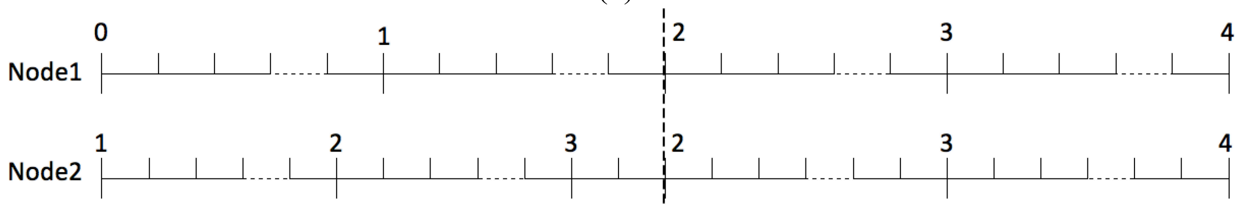
$$\lim_{t\to\infty} \hat{C}_i(t) = C_c(t) \tag{4}$$

Consensus clock is a virtual clock within the sensor network. For most WSN applications, there is no need for the clocks of sensor nodes to converge to the real clock. Take speed estimation as an example. As long as all sensor nodes are synchronized, they can successfully locate the target, and therefore successfully estimate the moving speed. In this case clock synchronization to a virtual clock is enough.

Figure 2. An example of offset and skew rate compensation. (**a**) Before clock synchronization; (**b**) After clock synchronization.



(**a**)



(**b**)

Figure 2 shows an example of offset and skew rate compensation. In Figure 2a, before clock synchronization, Node 1 and Node 2 have different offset and skew rates. For offset compensation, clock readings "2" for Node 1 and Node 2 are not aligned. For skew rate, it is equal to the multiplication of tick duration, T_{tick}^i, and counter limit, N_i, as shown in Equation (5). Either different T_{tick}^i or different N_i could make the two nodes have different α_i:

$$\alpha_i = N_i \times T^i_{tick} \tag{5}$$

After clock synchronization, as shown in Figure 2b, Node 2 is synchronized to Node 1. For offset compensation, the two nodes align the start time of the clock reading "2". For skew rate compensation, Node 2 adjusts its counter limit to make the clock duration equal to that of Node 1. It should be noted that only the counter limit can be adjusted by users. Tick duration is a natural property of the oscillator and therefore cannot be modified.

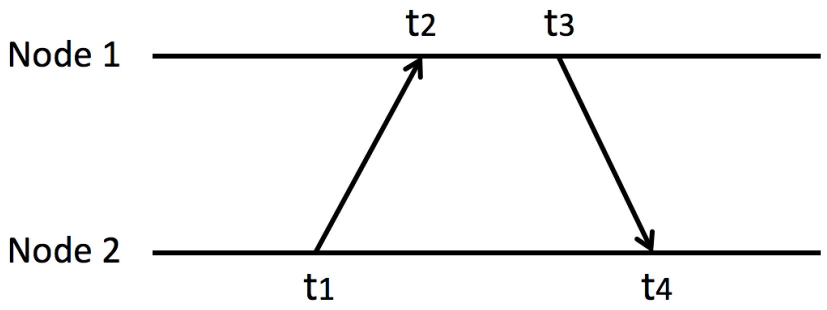
4. The Proposed GNA Algorithm for Clock Synchronization

The GNA algorithm includes two parts: offset compensation and skew rate compensation. They are realized by clock exchange and computation. The clocks of the sensor nodes will eventually converge to a consensus clock. One thing that should be noted for clock exchange among sensor nodes, is that the propagation delay incurs errors and this leads to inaccuracy. This will be compensated by two-way message exchange.

4.1. Propagation Delay Model

The signal transmission from one node to another experiences a propagation delay. This delay would cause the inaccuracy for clock synchronization between two nodes [15–17]. The idea for compensating this propagation delay is using two-way message exchange. As shown in Figure 3, Node 2 asks Node 1 for time. Node 2 records the time instant t_1, when it sends message. Node 1 receives the message and records the instant t_2. Then Node 1 returns the time information message and records the time instant t_3. Node 2 receives it and keeps the time instant t_4. Using these four time instants, Node 2 would estimate the propagation delay using Equation (6). Then it can compensate for the time offset. In the remainder of this paper, all the message exchanges adopt two-way message exchange. This is minimum information exchange for propagation delay estimation.

Figure 3. Two-way message exchange.



$$delay = \frac{(t4 - t1) - (t3 - t2)}{2} \tag{6}$$

4.2. Offset Compensation

Offset compensation is to make the sensor nodes to align at a common value of time. It is equivalent to obtaining $\hat{\beta}_i$ to satisfy Equation (4). We expand it as illustrated in Equation (7):

$$\begin{aligned}\lim_{t\to\infty}\frac{\alpha_i}{\hat{\alpha}_i} * t + \beta_i - \hat{\beta}_i &= C_c(t) \\ &= \alpha_c * t + \beta_c\end{aligned} \tag{7}$$

We have:

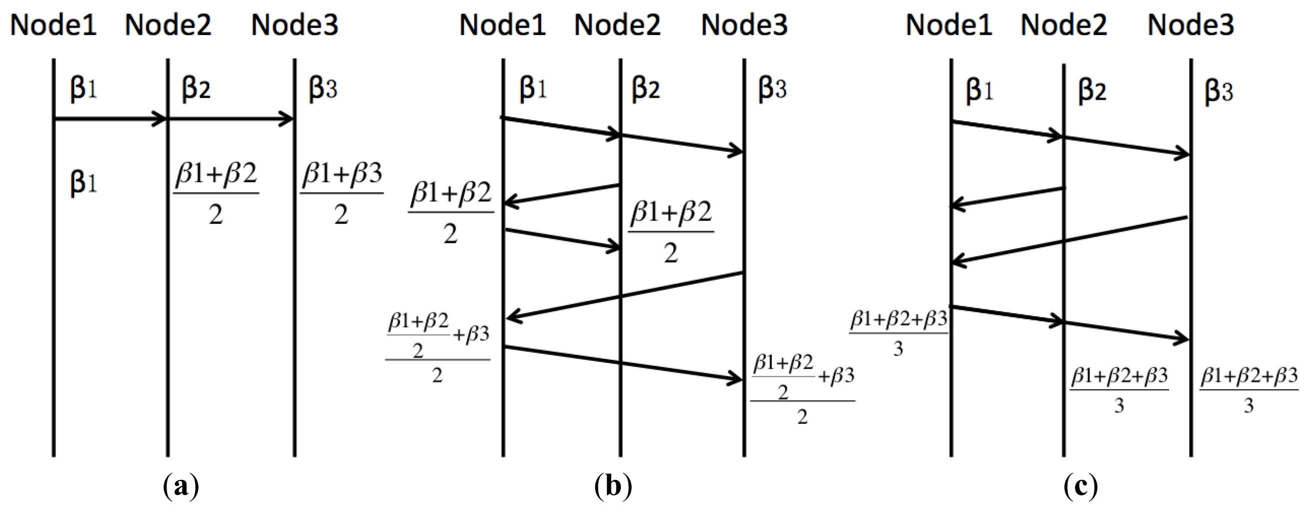
$$\hat{\beta}_i = \beta_i - \beta_c \tag{8}$$

Since sensor networks normally have a large number of sensor nodes covering a relatively big area, is impossible for a certain node to directly communicate with all the nodes in a single hop way. Therefore, local information exchange and averaging is used in this algorithm. The nodes take turns to collect time information from their neighbors and take an average of the collected time information and their own value. The result is estimated as the consensus value of the node and neighbors. The algorithm of offset compensation is presented as below:

Step 1: Periodically sense the channel, if the channel is clear, then broadcast a requesting packet to ask the neighbors within its transmission range for the clock information.
Step 2: The neighbors return the clock values.
Step 3: Compensate for the propagation delay
Step 4: The node calculates the average offset of all the neighbors as well as itself.
Step 5: Update its own offset value.
Step 6: Broadcast the compensation table. The table includes node ID and compensated offset.
Step 7: The neighbors receive the table and update their own clock offset.
Step 8: Repeat.

In this paper, 2-node offset averaging which is used in [19–22] is replaced by group neighborhood averaging with propagation delay compensation. As shown in Figure 4, it is assumed that Node 1 is the randomly selected node which synchronizes neighboring nodes. In Figure 4a, 2-node offset averaging is realized by broadcasting the synchronization message. Node 2 and Node 3 receive this message and calculate the average of the offsets of Node 1 and themselves. In this model, the propagation delay is ignored. The three nodes obtain different clock values and the deviation of errors becomes smaller. This method is used in [19–22]. In Figure 4b, two-way message exchange is adopted to compensate for the propagation delay between two nodes. Node 1 broadcasts a request message and records the time instant. Assume Node 2 replies earlier. Node 1 takes the average of Node 1 and Node 2's offsets. Then Node 1 sends the clock and propagation delay information to Node 2. Node 2 is synchronized with Node 1. Next Node 3 follows the same procedure. In this scenario, Node 1 needs to send different synchronization messages to every neighboring node, and these nodes do not converge to a single clock. In Figure 4c which is proposed by this paper, Node 1 waits to collect all the clock information from neighboring nodes. It calculates the propagation delay and takes the group average among all the nodes. In this way, an accurate, fast offset convergence is achieved. It should be mentioned that because there are three time message exchanges between the randomly elected node and its neighbor, one can use the first two message exchanges to estimate and compensate the propagation delay. There is no extra information exchange needed, therefore minimum information exchange is adopted for both propagation delay compensation and offset compensation.

Figure 4. Explanation of GNA algorithm. (**a**) 2-node synchronization without considering propagation delay; (**b**) 2-node synchronization with propagation delay compensation; (**c**) group neighborhood averaging with propagation delay compensation.



Next, it will be shown that the algorithm of offset compensation achieves global synchronization throughout the entire network. As defined in Section 2, β_i is the offset value of node *i*. It can be obtained as below:

Theorem 1. The offset values of all the sensor nodes in the network converge to certain value.

Proof. Assume that β_{max}^0 and β_{min}^0 are the maximum and minimum offset values among all the sensor nodes in the network at the initial time. β_{max}^n and β_{min}^n are the maximum value and minimum value among the offset values of all the sensor nodes after the *n*th round synchronization. For the (*n* + 1)th round of synchronization, each node conducts the group averaging within its transmission range. The maximum offset value would be averaged with some other offset values which are smaller than or equal to it. Then the result should be smaller than or equal to β_{max}^n. If this result is the new maximum offset value after the (*n* + 1)th round of synchronization, we can say $\beta_{max}^{n+1} \leq \beta_{max}^n$. If β_{max}^{n+1} is from the averaging operation group without β_{max}^n node, since the offset values in the group is smaller than or equal to β_{max}^n, we can also obtain $\beta_{max}^{n+1} \leq \beta_{max}^n$. β_{max}^{n+1} is equal to β_{max}^n only when all the offset values of the neighbors are equal to β_{max}^n. From β_{max}^0, to β_{max}^n, until $\beta_{max}^{+\infty}$, the maximum offset value is non-increasing over the whole synchronization time. Similarly, we have $\beta_{min}^{n+1} \geq \beta_{min}^n$, meaning that from β_{min}^0, to β_{min}^n, until $\beta_{min}^{+\infty}$, the minimum offset value is non-decreasing over the whole synchronization time. Because the averaging operation is conducted all the time, we can obtain that $\lim_{n\to+\infty} \beta_{max}^n = \lim_{n\to+\infty} \beta_{min}^n = \beta_c$. This demonstrates that the offset values of all the sensor nodes converge to the certain value β_c finally.

Next we discuss the relationship between β_c and the initial average offset value of all the sensor nodes β_{avg}^0. For each neighborhood averaging, the average value of this sensor group does not change. Similarly, for each round of synchronization, the average value of the entire network does not change. Then we have $\beta_{avg}^0 = \beta_{avg}^n = \beta_{avg}^\infty$. As discussed above, after certain rounds of synchronization, the offset values of all the sensors converge to β_c. Then the average offset value of these sensors would be

$\frac{1}{k}\sum_{i=1}^{k}\beta_c = \beta_c$. Therefore, we obtain $\beta_c = \beta_{avg}^{\infty} = \beta_{avg}^{0}$. The offset values of all the sensor nodes in the network converge to the initial average offset value.

MAC layer time stamping is adopted in the proposed protocol. It is well explained in detail in [19,22]. On the transmitter side, the timestamp is recorded into the message just before the packet is transmitted into the air at the physical layer. On the receiver side, once the message "Start Frame Delimiter" (SFD) is received, the instant time is recorded as the start of the packet reception. This mechanism significantly reduces the unpredictable delays in the protocol stacks at both transmitter and receiver sides. The computational complexity of the algorithm $\mathrm{T(n)} = \mathcal{O}(\mathrm{n})$, where n is the number of nodes within the transmission range of a sensor node.

4.3. Skew Rate Compensation

Offset compensation is for time alignment among the network nodes. The skew rate also needs to be synchronized so that the sensor nodes can have a common clock speed. The method is the same as offset compensation by taking average of the skew rate values among the sensor nodes as illustrated in Equation (9):

$$\hat{\alpha}_i = \frac{1}{N}\sum_{i=1}^{N}\alpha_i \tag{9}$$

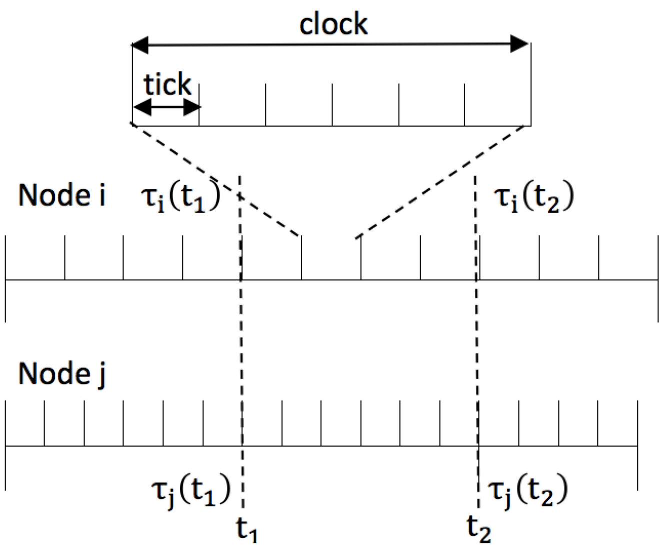
The algorithm of skew rate compensation is given below:

Step 1: Periodically sense the channel, if channel is clear, then broadcast a requesting packet to ask the neighbors within its transmission range for the skew rate.
Step 2: The neighbors return the skew rate.
Step 3: Compensate for the propagation delay
Step 4: The node calculates the average skew rate for all these nodes of the neighbors and itself.
Step 5: Update its skew rate value.
Step 6: Broadcast the compensation table. The table includes node ID and compensated skew rate.
Step 7: The neighbors receive the table and update their own skew rates.
Step 8: Repeat.

The skew rates of sensor nodes in the network will converge finally to a certain value. The proof is the same as Theorem 1. The computational complexity of the algorithm $T(n) = \mathcal{O}(n)$, where n is the number of nodes within the transmission range of a sensor node. This is the same as the reference protocols, FWA and CWA.

To illustrate the skew rate compensation clearly, we take an example of two-node skew rate compensation. We collect the clock readings $\tau_i(t_1)$ and $\tau_j(t_1)$ at the absolute time t_1 from node *i* and node *j*, and the clock readings $\tau_i(t_2)$ and $\tau_j(t_2)$ at the absolute time t_2 as shown in Figure 5 (clock readings collected at the same time are achieved by delay estimation mentioned in Part A in this section. The algorithm demonstrates that there is a two-way message exchange). N_i and N_j are the maximum limit of the clocks of node *i* and node *j*. When the counter reaches the maximum limit, an interrupt is triggered and the clock value increments. The relationship of the clock duration T_{clock}, tick duration T_{tick} and counter limit *N* is given in Equation (10).

Figure 5. An example of skew rate compensation.



$$T_{clock} = N \times T_{tick} \tag{10}$$

For skew rate synchronization, based on the pre-defined counter limits N_i, N_j and the collected clock readings $\tau_i(t_1)$, $\tau_j(t_1)$, $\tau_i(t_2)$ and $\tau_j(t_2)$, the goal is to find N_j^+, which is the compensated counter limit of node *j*, to satisfy $T_{clock}^i = T_{clock}^j{}^+$.

For node *i*, the duration between the two probe messages, $t_2 - t_1$, can be expressed as Equation (11). Similarly, $t_2 - t_1$ can also be expressed from node *j* as shown in Equation (12):

$$t_2 - t_1 = \left(\tau_i(t_2) - \tau_i(t_1)\right) \times N_i \times T_{tick}^i \tag{11}$$

$$t_2 - t_1 = \left(\tau_j(t_2) - \tau_j(t_1)\right) \times N_j \times T_{tick}^j \tag{12}$$

Combining Equations (11) and (12), we have:

$$\frac{\tau_i(t_2) - \tau_i(t_1)}{\tau_j(t_2) - \tau_j(t_1)} = \frac{N_j \times T_{tick}^j}{N_i \times T_{tick}^i} = \frac{T_{tick}^j}{N_i \times T_{tick}^i} \times N_j \tag{13}$$

Since the goal is to satisfy $T_{clock}^i = T_{clock}^j{}^+$, we expand it as Equation (14):

$$N_i \times T_{tick}^i = N_j^+ \times T_{tick}^j \tag{14}$$

Combining Equations (13) and (14), the new counter limit of node *j*'s oscillator is obtained in Equation (15):

$$N_j^+ = \frac{\tau_j(t_2) - \tau_j(t_1)}{\tau_i(t_2) - \tau_i(t_1)} \times N_j \tag{15}$$

The above discussion is to synchronize node *j* to node *i*. If the average compensation is desired, then the goal becomes to find N_i^+ and N_j^+, which satisfies Equation (16):

$$T_{clock}^i{}^+ = T_{clock}^j{}^+ = \frac{T_{clock}^i + T_{clock}^j}{2} \tag{16}$$

Since $T_{clock} = N \times T_{tick}$, for node *i* we have:

$$T_{tick}^{i} \times N_i^{\ +} = \frac{N_i \times T_{tick}^{i} + N_j \times T_{tick}^{j}}{2} \tag{17}$$

By dividing T_{tick}^{i} on both sides of Equation (17) and combining it with Equation (13), we obtain:

$$N_i^{\ +} = \frac{1 + \frac{\tau_i(t_2) - \tau_i(t_1)}{\tau_j(t_2) - \tau_j(t_1)}}{2} \times N_i \tag{18}$$

Similarly:

$$N_j^{\ +} = \frac{1 + \frac{\tau_j(t_2) - \tau_j(t_1)}{\tau_i(t_2) - \tau_i(t_1)}}{2} \times N_j \tag{19}$$

So far, the skew rates are compensated to a consensus value for the two nodes by adjusting the counter limit. This method can be extended to more nodes. After a certain number of compensation rounds, the entire network would eventually be synchronized. It should be noted that for neighboring information exchange, it is a two-way communication. By using the recorded clock values, the propagation delay can be estimated. Using the estimated propagation delay value, the clock readings of different nodes at the same time, e.g., $\tau_i(t_1)$ and $\tau_j(t_1)$, can be obtained. In Section 4 the algorithms are evaluated by simulations and the results are analyzed.

5. Simulation Evaluation

The algorithm of GNA clock synchronization has been simulated by MATLAB and compared with forward weighted average (FWA) and confidence weighted average (CWA) which are proposed in [20]. For FWA, Each node periodically broadcasts a synchronization beacon. The receiving node takes average of offset and drift. FWA calculates the average only between two nodes, as expressed as Equation (20). $\hat{C}_i(t)$ is the estimation of $C_i(t)$. $C_i(t)$ and $C_j(t)$ are the clock readings from node *i* and node *j*. CWA is the improved algorithm based on FWA. CWA includes a confidence parameter which gives more weighting to the nodes that conduct more rounds of synchronization. It is expressed in Equation (21). The initial values of γ_i and γ_j are set to 1:

$$\hat{C}_i(t) = \frac{C_i(t) + C_j(t)}{2} \tag{20}$$

$$\gamma_i = \gamma_i + 1 \tag{21}$$

A 10 × 10 grid sensor network is built up. After two minutes, the synchronization procedure starts and it performs the algorithm every minute for ten rounds. The simulation parameters are described in Table 1. Several cases are discussed to evaluate the proposed GNA algorithm.

Table 1. Simulation parameters.

Parameters	Values	Parameters	Values
Network size	10 × 10	Initial offset	−1000 ~ +1000
Synchronization rounds	10	Initial relative skew rate	<20 ppm
Initial sync time	2	Algorithms	FWA, CWA, GNA
Transmission range	1	Multiple runs	100

Case 1: $\alpha_i = 1$, $\beta_i \neq 0$

In Case 1, α_i is equal to 1, while β_i is not equal to 0. The skew rates of all the nodes in the network are always the same. An initial error following normal distribution is given to all the sensor nodes. Figure 6 shows the geographic distribution of synchronization error before and after the first round of offset compensation. It can be seen that before the offset synchronization, the range of clock error is from −600 to +600 ticks. After one round of synchronization, the range of clock error reduces to −20 to +40 ticks. The offset is significantly compensated. It is also noted that after the offset synchronization, the values are the same for several node clusters. This is because the average algorithm is conducted for all the nodes within the transmission range of a central node. Then after the average calculation for one cluster, the nodes within the cluster would have a common clock.

Figure 6. Geographic distribution of synchronization error before and after the first round of offset compensation.

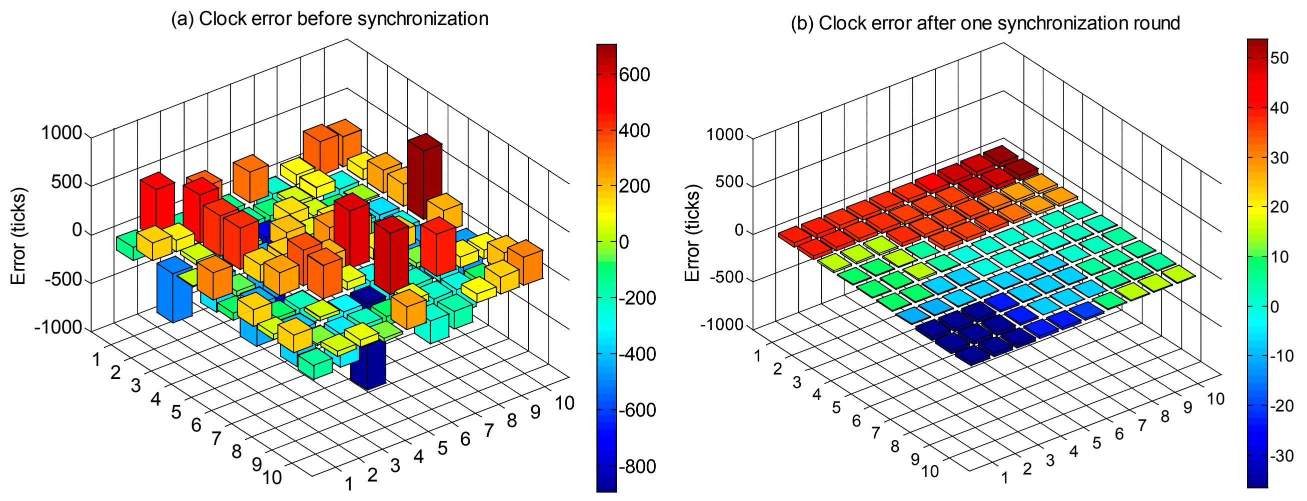


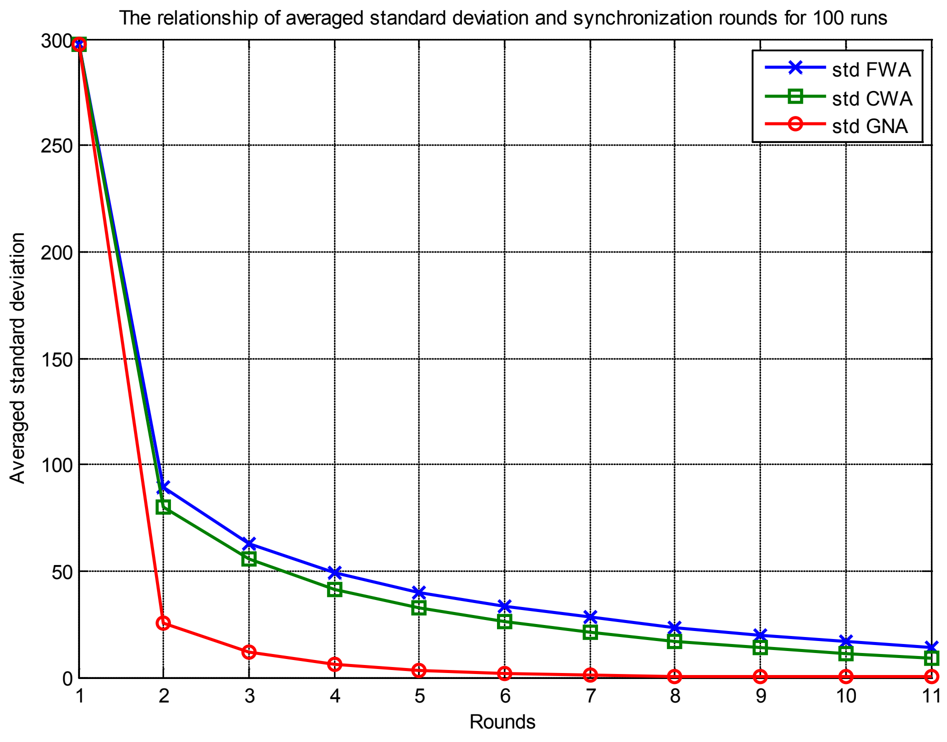
Figure 7 shows the relationship of the averaged standard deviation of errors and the synchronization rounds for 100 runs for three clock synchronization algorithms: FWA, CWA and GNA. For each run, random initial errors are generated for each sensor node. The standard deviation is calculated as in Equation (22):

$$std = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{22}$$

where: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

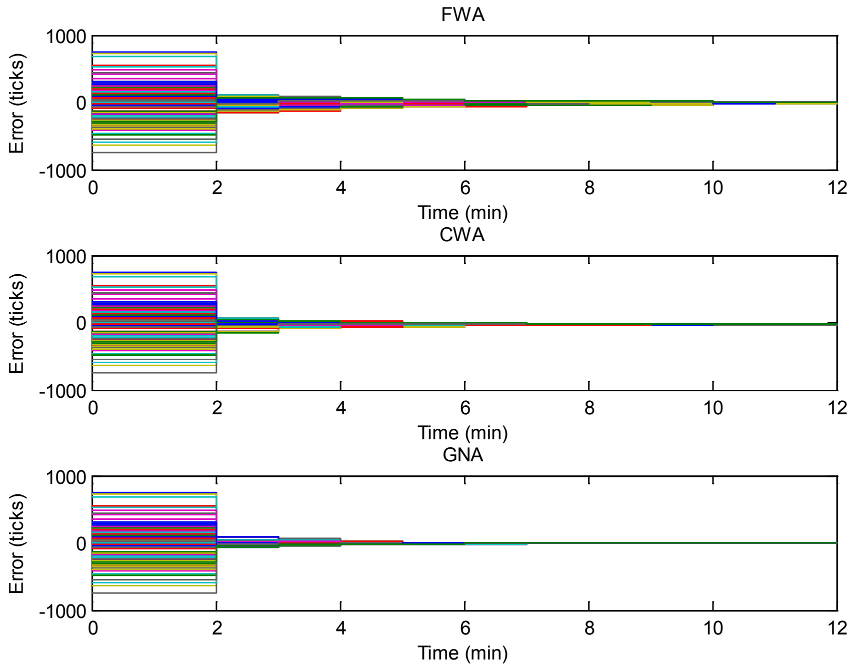
After each round of synchronization, the standard deviation of clock offsets for a specific clock synchronization protocol is calculated and plotted. The standard deviation expresses the convergence performance by calculating how much variation exists from the average. A low standard deviation indicates that the errors of the sensor nodes tend to be very close to the mean, while a high standard deviation indicates that the errors are spread out over a large range of values. From Figure 7 it can be seen that the averaged standard deviation of the errors in the beginning is 290 ticks. It decreases as the synchronization is conducted round by round. As the synchronization is conducted, the standard deviation reduction trend becomes slow and the value goes towards zero. This is realized by the averaging behavior for the three clock synchronization algorithms. The algorithms average the clock offsets between two nodes or among a number of sensor nodes, therefore the standard deviation reduces. The performance of the proposed GNA protocol is better than that of FWA and CWA. For each round of clock synchronization, GNA achieves a lower standard deviation than FWA and CWA. This is because the offset averaging calculation is conducted among all the neighbor nodes and the obtained value is assigned to these sensor nodes when using GNA algorithm, but for FWA and CWA the average is taken only between a pair of nodes for each round. The speed of convergence of FWA and CWA is slower than that of GNA.

Figure 7. Averaged standard deviation of FWA, CWA and GNA for 10 rounds of clock synchronization.



The case of the errors in ticks of the FWA, CWA and GNA algorithm for 10 rounds of synchronization are shown in Figure 8. For each round of clock synchronization, the offsets are compensated. Because the skew rates in this case are set to 1, the tick errors would last until the next clock synchronization. The GNA algorithm achieves the fastest convergence among the three algorithms. The error converges to zero after two rounds of clock synchronization (3 min after the simulation). Zero means that the sensor nodes converge to a common virtual clock. FWA and CWA need a longer time to converge. From the figure it can be seen that FWA and CWA do not converge to zero even after 10 rounds of clock synchronization. As discussed above, the better performance of GNA is because GNA takes average of all neighbor nodes but FWA and CWA only calculate the average between two nodes.

Figure 8. Errors of FWA, CWA and GNA with only offset compensation but without skew rate compensation.



Case 2: $\alpha_i \neq 1$, $\beta_i = 0$

In Case 2, there is no initial offset given to the sensor nodes in the network. Therefore, β_i is equal to 0, but α, the skew rate of the sensor nodes clock is assigned a random value, which is not equal to 1, for each sensor node. The off-center skew rate causes clock errors. The clock errors for FWA, CWA and GNA algorithms within 12 min are compared in Figure 9. In this figure, only offsets are compensated, while the skew rates are not compensated. From the figure there are no initial errors in the first two minutes of the simulation. After that a random skew rate is generated for all the sensor nodes. Due to this skew rate, the errors increase in the next minute. At the third minute of the simulation, the three algorithms conduct the offset compensation. The clock errors of the sensor nodes are reduced, but because there is no skew rate compensation, the clock errors increase again in the next minute. The clock error patterns come out periodically. From the figure, for each clock synchronization at the integral time instant, the proposed GNA outperforms than FWA and CWA by obtaining a smaller clock error deviation of all the sensor nodes for each round of synchronization. As mentioned above, group averaging is the reason for GNA's better performance.

Figure 10 gives the simulation result of the comparisons of FWA, CWA and GNA for both offset and skew rate compensation. At the time instant of 2 min random skew rates are initialized for all the sensor nodes. The clock errors increase in the next minute. At the time instant of 3 min, The FWA, CWA and GNA algorithms are executed for both offset compensation and skew rate compensation. The clock errors therefore become smaller. Since one round of skew rate compensation cannot make all sensor nodes achieve a common skew rate, the clock errors still deviate in the following minutes. Then the next round of synchronization comes. The offsets and skew rates of sensor nodes are compensated again. From the figure it can be seen that at the end of the simulation, the clock errors of all the three algorithms are synchronized to a virtual consensus clock which is already not the real time.

Figure 9. Errors of FWA, CWA and GNA with only offset compensation but without skew rate compensation.

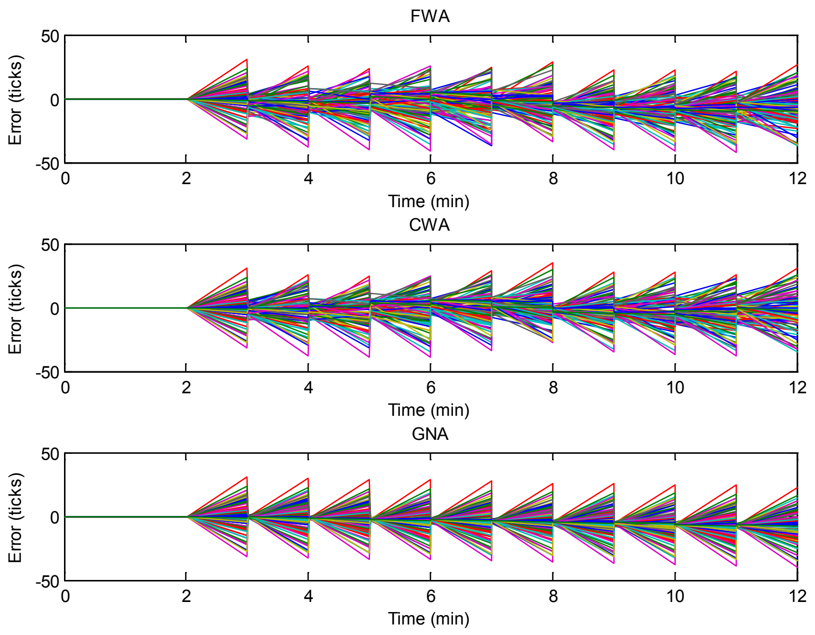
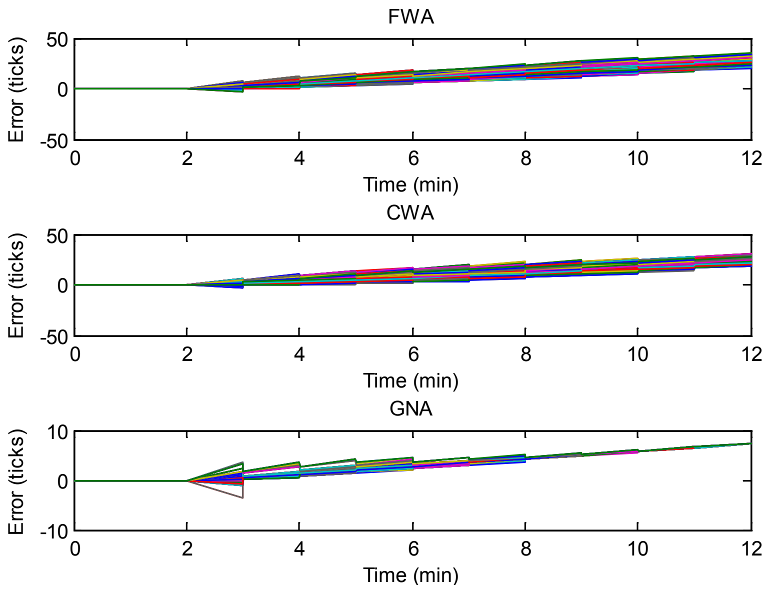


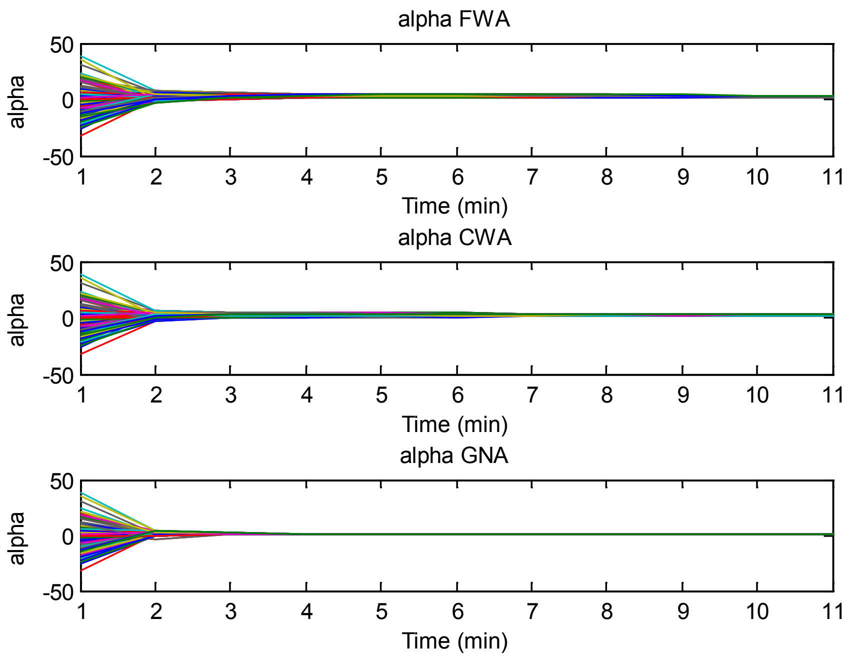
Figure 10. Errors of FWA, CWA and GNA of both offset compensation and skew rate compensation.



This is the nature of the consensus algorithm. The consensus algorithm can make the clocks of sensor nodes converge, but cannot guarantee that they will converge to the real time since there is no reference clock in the network. However, this is not an issue for most applications which require only internal clock synchronization. The convergence performance of GNA is much better than that of FWA and CWA. The two-node synchronization cannot make the clock converge.

In Figure 11, the skew rates of the proposed GNA, FWA and CWA are compared. Initial skew rates are generated at the beginning of the simulation. After 10 rounds of synchronization, the deviation of the skew rate becomes smaller. The proposed GNA converges faster than FWA and CWA. After six rounds of synchronization, the skew rate of GNA converges to a common value, but for FWA and CWA, the skew rates do not converge to a common value even after 10 rounds of synchronization. Group averaging within the transmission range is still the reason for the fast convergence of GNA.

Figure 11. The skew rate of FWA, CWA and NA for the 10 round clock synchronizations.



Case 3: $\alpha_i \neq 1$, $\beta_i \neq 0$

In Case 3, both initial errors and skew rates are assigned to all the sensor nodes. The errors in ticks of FWA, CWA and GNA algorithm for 10 round synchronizations are shown in Figure 12. In this simulation both offset and skew rate compensation are conducted for FWA, CWA and GNA. The initial clock errors are assigned to the sensor nodes in the beginning. After two minutes, the first round of offset and skew rate compensation are conducted. The clock errors decrease. During the next minute, due to the existence of skew rate deviation, the clock errors increase. In this figure this clock error increase cannot be clearly seen because the increase is not that significant compared with the initial errors with the range of −1000 to 1000 ticks. After 10 rounds of offset and skew rate compensation, the clocks of the sensor nodes in the network converge. It can be seen that GNA converges faster than FWA and CWA.

Figure 13 shows the averaged standard deviation of clock errors for FWA, CWA and the proposed GNA algorithm during the 10 rounds of offset and skew rate compensation for 100 runs. For each run random initial errors are generated for all the sensor nodes. This result indicates a similar explanation as in Figure 7. At the beginning of the simulation, the standard deviation of the clock errors of the sensor nodes are big, here around 300 ticks. After one round of synchronization, the standard deviation drops significantly to around 50. As time collapses, the standard deviation continues to decrease towards zero. This illustrates that the algorithms of FWA, CWA and the proposed GNA are effective for the purpose of clock synchronization for WSNs. From the figure it can also be seen that the proposed GNA achieves better performance of convergence than FWA and CWA.

Figure 12. Errors of FWA, CWA and GNA of both offset compensation and skew rate compensation with initial offset and skew rate errors.

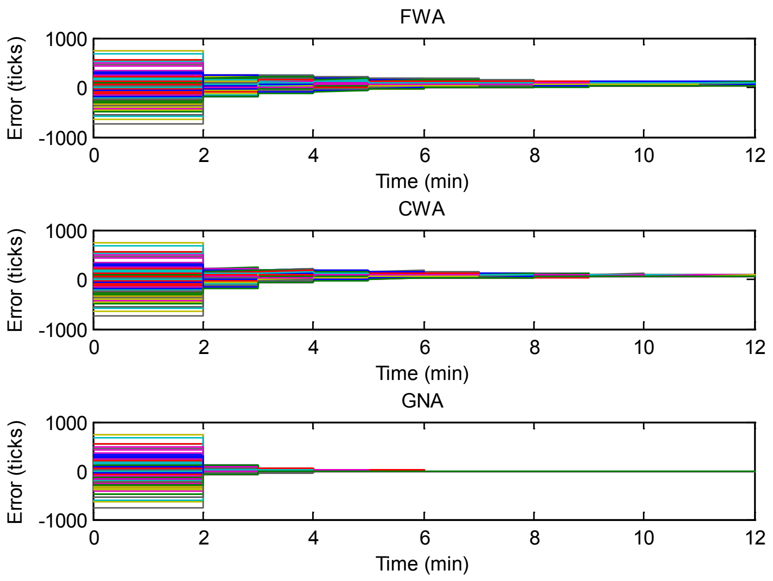


Figure 13. Averaged standard deviation of FWA, CWA and GNA for 10 rounds of synchronization.

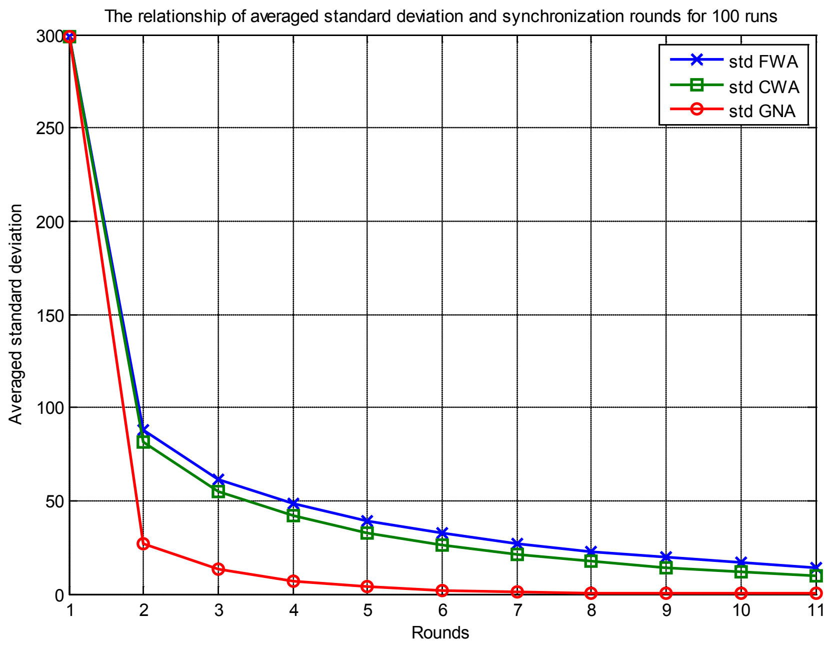


Figure 14 shows the averaged standard deviation of errors for different transmission ranges using the proposed GNA for 100 runs. Transmission ranges equal to 1, 2 and 3 unit length are simulated. From the figure when the transmission range is equal to 1, the averaged standard deviation in the beginning is 28 ticks. As time collapses, the standard deviation decreases significantly. After ten rounds of synchronization, the standard deviation drops to zero. When the transmission range increases to 2 or 3, the performance of convergence becomes significantly better. This is because when the transmission range increases, more nodes are involved for the average calculation. *N* in Equations (6) and (7) becomes larger. The convergence becomes faster throughout the entire network.

Figure 14. Averaged standard deviation of errors for different transmission range using GNA.

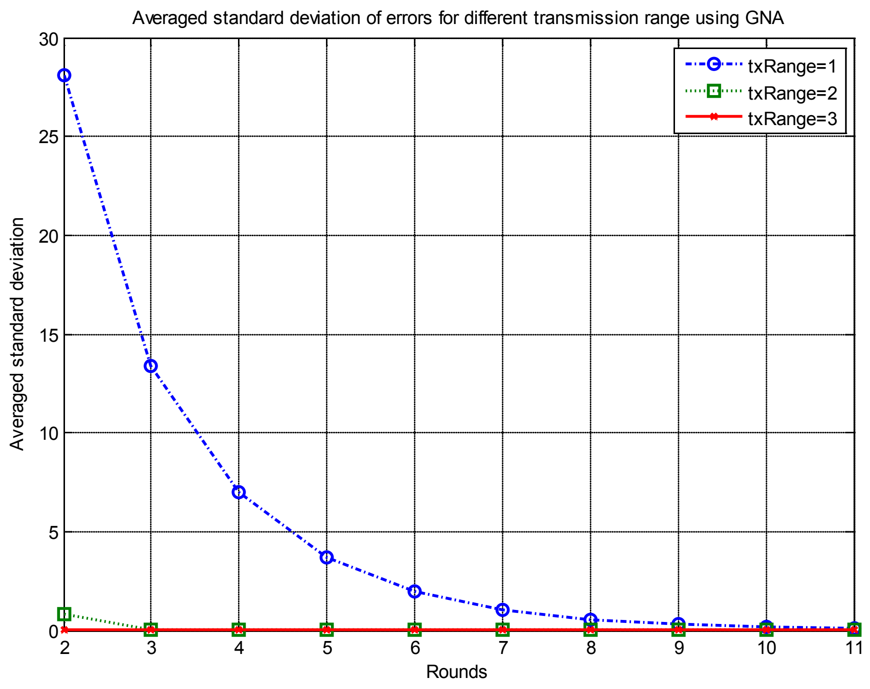


Figure 15. Averaged standard deviation of errors for different network sizes using GNA.

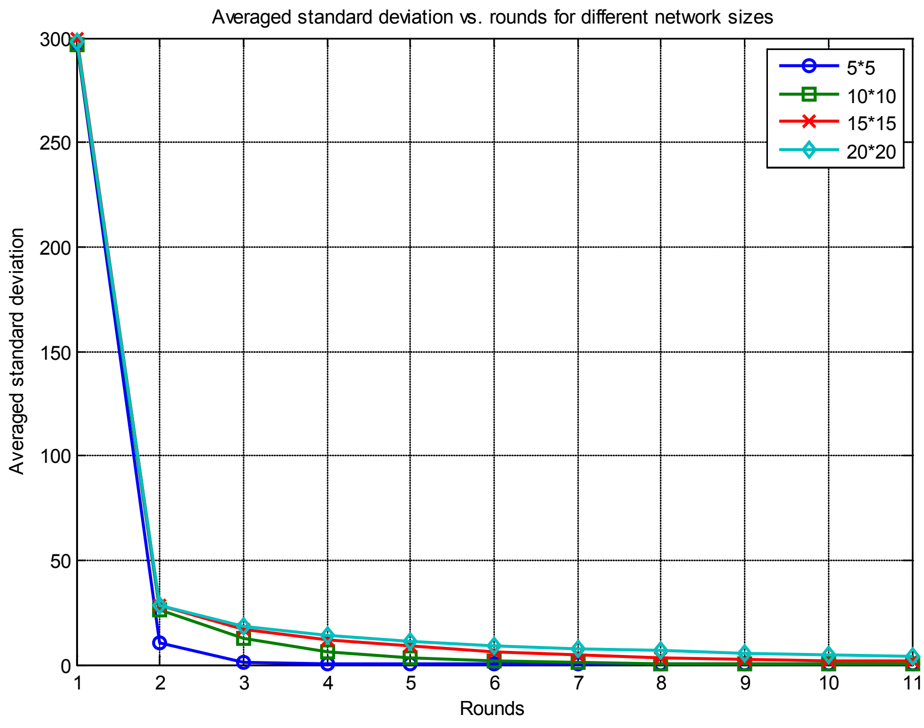


Figure 15 shows the averaged standard deviation of errors *vs.* number of rounds for different network sizes using the GNA algorithm. There are four scenarios: 5 × 5, 10 × 10, 15 × 15 and 20 × 20. The standard deviation of errors goes down as the synchronization iteration increases. This demonstrates the convergence of the proposed GNA algorithm. It can also be seen from the figure that the standard deviation of errors for bigger network size is larger than that for smaller network size. The reason is that the transmission range is limited. The averaging operation can only be taken within a small cluster. If the network size is larger, meaning that more sensor nodes are in the network, of course after the same rounds of synchronization, the standard deviation of errors is bigger. The convergence rate is slower if the network size increases. Therefore, more rounds of synchronization are needed for global synchronization if the network size becomes larger.

6. Conclusions

This paper has proposed a novel internal clock synchronization solution for wireless sensor networks. It is a fully distributed, network-wide synchronization algorithm. Compared with several recent popular synchronization algorithms, the proposed GNA algorithm considers the propagation delay. The two-way message exchange is adopted for the compensation of the propagation delay. Based on this new coming message exchange, a group average scheme was proposed. The random elected sensor node collects the offset and skew rate from neighbors and takes group averaging, which would achieve fast clock convergence. The analytical analysis of offset and skew compensation is described. The performance of the proposed GNA is fully simulated and analyzed for different kinds of offset and skew rate compensation scenarios. The simulation results showed that the proposed GNA algorithm achieves better performance in terms of accuracy and convergence speed than other algorithms like FWA and CWA. Future work would focus on full mathematical interpretation of clock synchronization in WSNs, energy consumption analysis and hardware testbed implementation.

Acknowledgments

This study has been supported by Science and Technology Commission of Shanghai Municipality (STCSM) under the grant No. 13510500400.

Author Contributions

The corresponding author Lin Lin, who is responsible for the overall work, proposed the research idea, conducted the simulation and prepared the manuscript. The second author Shiwei Ma has provided general guidance during the whole research. The third author Maode Ma has given quite a lot of comments and suggestions for the simulations and paper writings.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Stoianov, I.; Nachman, L.; Madden, S.; Tokmouline, T.; Csail, M. PIPENET: A Wireless Sensor Network for Pipeline Monitoring. In Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN), Cambridge, MA, USA, 25–27 April 2007; pp. 264–273.
2. Makoto, S.; Shunsuke, S.; Narito, K.; Hiroyuki, M. A high-density earthquake monitoring system using wireless sensor networks. In Proceedings of the 5th International Conference on Embedded Networked Sensor Systems, Sydney, Australia, 4–9 November 2007; pp. 373–374.

3. Torfs, T.; Sterken, T.; Brebels, S.; Santana, J.; van den Hoven, R.; Spiering, V.; Bertsch, N.; Trapani, D.; Zonta, D. Low Power Wireless Sensor Network for Building Monitoring. *IEEE Sens. J.* **2013**, *13*, 909–915.
4. Kim, S.; Pakzad, S.; Culler, D.; Demmel, J.; Fenves, G.; Glaser, S.; Turon, M. Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks. In Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN), Cambridge, MA, USA, 25–27 April 2007; pp. 254–263.
5. Mainwaring, A.; Culler, D.; Polastre, J.; Szewczyk, R.; Anderson, J. Wireless sensor networks for habitat monitoring. In Proceedings of the International Workshop on Wireless Sensor Networks and Applications, Atlanta, GA, USA, 28 September 2002; pp. 88–97.
6. Otal, B.; Alonso, L.; Verikoukis, C. Highly reliable energy-saving MAC for wireless body sensor networks in healthcare systems. *IEEE J. Sel. Areas Commun.* **2009**, *27*, 553–565.
7. Lin, L.; Wong, K.-J.; Kumar, A.; Tan, S.L.; Phee, S.J. An energy efficient MAC protocol for mobile *in vivo* body sensor networks. In Proceedings of the 2011 Third International Conference on Ubiquitous and Future Networks (ICUFN), Dalian, China, 15–17 June 2011; pp. 95–100.
8. Wu, Y.-C.; Chaudhari, Q.; Serpedin, E. Clock synchronization of wireless sensor networks. *IEEE Signal Process. Mag.* **2011**, *28*, 124–138.
9. Sivrikaya, F.; Yener, B. Time synchronization in sensor networks: A survey. *IEEE Netw.* **2004**, *18*, 45–50.
10. Sundararaman, B.; Buy, U.; Kshemkalyani, A.D. Clock synchronization for wireless sensor networks: A survey. *Ad Hoc Netw.* **2005**, *3*, 281–323.
11. Faizulkhakov, Y.R. Time synchronization methods for wireless sensor networks: A survey. *Program. Comput. Soft.* **2007**, *33*, 214–226.
12. Mills, D.L. Internet time synchronization: The network time protocol. *IEEE Trans. Commun.* **1991**, *39*, 1482–1493.
13. Jeremy, E.; Lewis, G.; Deborah, E. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.* **2002**, *36*, 147–163.
14. Sichitiu, M.L.; Veerarittiphan, C. Simple, accurate time synchronization for wireless sensor networks. In Proceedings of the IEEE Wireless Communications and Networking, New Orleans, LA, USA, 20 March 2003; pp. 1266–1273.
15. Ganeriwal, S.; Kumar, R.; Srivastava, M.B. Timing-sync protocol for sensor networks. In Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, Los Angeles, CA, USA, 5–7 November 2003; pp. 138–149.
16. Akhlaq, M.; Sheltami, T.R. RTSP: An Accurate and Energy-Efficient Protocol for Clock Synchronization in WSNs. *IEEE Trans. Instrum. Meas.* **2013**, *62*, 578–589.
17. Maroti, M.; Kusy, B.; Simon, G.; Ledeczi, A. The flooding time synchronization protocol. In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, Baltimore, MD, USA, 3–5 November 2004; pp. 39–49.
18. Su, W.; Akyildiz, I.F. Time-diffusion synchronization protocol for wireless sensor networks. *IEEE/ACM Trans. Netw.* **2005**, *13*, 384–397.
19. Schenato, L.; Fiorentin, F. Average TimeSynch: A consensus-based protocol for clock synchronization in wireless sensor networks. *Automatica* **2011**, *47*, 1878–1886.

20. Maggs, M.K.; O'Keefe, S.G.; Thiel, D.V. Consensus Clock Synchronization for Wireless Sensor Networks. *IEEE Sens. J.* **2012**, *12*, 2269–2277.
21. Li, Q.; Rus, D. Global clock synchronization in sensor networks. *IEEE Trans. Comput.* **2006**, *55*, 214–226.
22. Sommer, P.; Wattenhofer, R. Gradient clock synchronization in wireless sensor networks. In Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN), Washington, DC, USA, 23 October 2009; pp. 37–48.
23. Zhao, D.; An, Z.; Xu, Y. Time Synchronization in Wireless Sensor Networks Using Max and Average Consensus Protocol. *Int. J. Distrib. Sens. Netw.* **2013**, *2013*, 192128.
24. Gang, X.; Shalinee, K. Analysis of distributed consensus time synchronization with Gaussian delay over wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.* **2009**, doi:10.1155/2009/528161.
25. Leng, M.; Wu, Y.-C. Distributed clock synchronization for wireless sensor networks using belief propagation. *IEEE Trans. Signal Process.* **2011**, *59*, 5404–5414.
26. Du, J.; Wu, Y. Distributed Clock Skew and Offset Estimation in Wireless Sensor Networks: Asynchronous Algorithm and Convergence Analysis. *IEEE Trans. Wirel. Commun.* **2013**, *12*, 5908–5917.