sensors

Article

Testing a Firefly-Inspired Synchronization Algorithm in a Complex Wireless Sensor Network

Chuangbo Hao, Ping Song *, Cheng Yang and Xiongjun Liu

Key Laboratory of Biomimetic Robots and Systems (Ministry of Education), Beijing Institute of Technology, Beijing 100081, China; 3120130073@bit.edu.cn (C.H.); ycthink@gmail.com (C.Y.); 3120140085@bit.edu.cn (X.L.)
* Correspondence: sping2002@bit.edu.cn; Tel.: +86-136-6136-5650

Academic Editors: Maria Gabriella Xibilia, Alex Alexandridis and Elias N. Zois
Received: 5 December 2016; Accepted: 5 March 2017; Published: 8 March 2017

Abstract: Data acquisition is the foundation of soft sensor and data fusion. Distributed data acquisition and its synchronization are the important technologies to ensure the accuracy of soft sensors. As a research topic in bionic science, the firefly-inspired algorithm has attracted widespread attention as a new synchronization method. Aiming at reducing the design difficulty of firefly-inspired synchronization algorithms for Wireless Sensor Networks (WSNs) with complex topologies, this paper presents a firefly-inspired synchronization algorithm based on a multiscale discrete phase model that can optimize the performance tradeoff between the network scalability and synchronization capability in a complex wireless sensor network. The synchronization process can be regarded as a Markov state transition, which ensures the stability of this algorithm. Compared with the Miroll and Steven model and Reachback Firefly Algorithm, the proposed algorithm obtains better stability and performance. Finally, its practicality has been experimentally confirmed using 30 nodes in a real multi-hop topology with low quality links.

Keywords: wireless sensor networks; firefly-inspired algorithm; coupling strength; multiscale model; stochastic coupling

1. Introduction

Data acquisition is the foundation of soft sensor and data fusion. In industrial processes, data often comes from multiple sources, which may be separated physically or virtually, and need to be synchronized. Therefore, distributed data acquisition and its synchronization are the important technologies to ensure the accuracy of soft sensors [1,2]. At present, Wireless Sensor Networks (WSNs), as an important distributed data acquisition technology, have been widely used in industry [3–5]. Synchronization is a key feature of WSNs, and it has been used in many wireless protocols. In general, the synchronization algorithm can be classified according to the centrality. On one hand, the centralized algorithm (e.g., Reference Broadcasts Synchronization [6], Timing-sync Protocol [7], and Flooding Time Synchronization Protocol [8]) sets a base time for the coordinator and the nodes modify their time correspondingly, thus the accuracy of this algorithm is high for structures with simple topology [9]. However, if the topological structure is complex, the centrality algorithm is inapplicable. On the other hand, compared with the centralized algorithm for synchronization, the distributed algorithm, which avoids single points of failure and is more robust, is more suitable for large-scale complex topological structures [10].

The distributed algorithm can be implemented with the classical packet-coupling synchronization or the firefly-inspired synchronization. The packet-coupling synchronization has accuracies ranging from microseconds to milliseconds [11]. However, a large number of packets need to be exchanged, which leads to increased computational complexity, energy expenditure, and poor scalability. Development of synchronization technology has led to the firefly-inspired synchronization [12–14].

It can obviate the drawbacks of packet-coupling methods emerging from pulse coupling. To date, most studies have relied on simulating firefly-inspired synchronization [15–19], because it is a physical layer-based method and depends heavily on the network hardware. Because firefly-inspired synchronization is not compatible with most commercially available WSN chips [20], very few groups have used firefly-inspired synchronization. Some groups have adopted firefly-inspired synchronization on special platforms such as a Wire [21,22], Ultra Wideband (UWB) [23], Custom Radio [24] and Light Emitting Diode (LED) [25].

To implement the firefly-inspired synchronization algorithm on more hardware platforms, this algorithm has recently been transformed into a software algorithm on a low Media Access Control (MAC) layer combined with packet-coupling synchronization [11], and is called the Reachback Firefly Algorithm (RFA). Based on the Miroll and Steven model (M&S model) [26], Werner-Allen et al. first utilized RFA with realistic radio effects and theoretically confirmed its stability [20]. The algorithm was tested on a 24-node indoor WSN and it was found that 50% of the nodes achieved synchronization within 154 μs [20]. However, the selection of the coupling strength (also called coupling factor, the definition can be found in [27]) and a random avoid time (the random transmission delay to node firing messages for avoiding repeated collisions) were not considered, which may influence the stability. Rather, the algorithm was simulated within a defined coupling strength range. Leidenfrost improved the theory of RFA and suggested the scope of avoid time and coupling strength in an all-to-all network where each node can communicate with every other node. However, in a complex topology, the time spread only reached milliseconds [27]. The refractory period is often assigned as a duration of time right after firing, during which no signal can be received from other nodes. Cui found that it had a positive effect on the convergence in RFA, however the design of the coupling strength was not taken into consideration [28]. A biologically inspired approach for distributed slot synchronization similar to RFA was implemented on a MAC layer and achieved slot synchrony in slotted communication systems by Tyrrell [29]. However, the synchronization was not tested on a realistic WSN and the design of the coupling strength was not considered. The simulation results showed that a coupling strength that was too high or low would lead to instability of the synchronization. Therefore, the coupling strength is still an open question for WSNs with complex topology, and it is difficult to select an appropriate coupling strength in a complex or a changeable topology. This problem is regarded as one of the most significant in the field of firefly-inspired algorithms [30]. The conflicts between synchronization messages have been present since synchronization was first developed [20,27]. Some researchers use a random avoid slot time to ease the conflicts, and the range of the avoid slot may affect the stability [20,27]. However, the range of the avoid slot was presented only for an all-to-all network, which is not suitable for a realistic WSN [27]. Hence, the conflicts between synchronization messages in a high-density network are still unresolved.

The focus of this paper is reducing the design difficulty of the firefly-inspired synchronization algorithm and to quantify the performance tradeoff between network scalability and synchronization capability in complex WSNs. We develop the discrete multiscale phase dynamics and design a stochastic coupling algorithm. The dynamics transform the phase into a state vector according to different quantitative levels. A node in the WSN broadcasts its state vector stochastically, which is then adjusted after receiving a neighboring state vector from a low MAC layer. The phase adjustment happens on each quantitative level with a constant adjustable step size, which not only ensures stability according to the global convergence property of Markov chains [31] without changing the coupling strength, but also achieves a more accurate and faster synchronization. In this paper, frequency drift and communication delay are also considered. The stability of the proposed algorithm is proven theoretically, and confirmed by comparing simulations with RFA in a complex WSN. It is found that the proposed algorithm achieves better performance. Finally, the algorithm is tested in a realistic, complex WSN with 30 nodes and achieves synchronization within 50 μs. Compared with the M&S model and RFA, this method has the following advantages:

1. By using a discrete state vector instead of a continuous phase, the algorithm can be easily implemented on a Microcontroller Unit (MCU) in a wireless module.
2. The step size can be adjusted, which ensures the stability according to the global convergence property of Markov chains without adjusting the coupling strength.
3. A better performance emerges from the stochastic coupling at multiscale quantitative levels and the channel congestion is alleviated significantly.

The rest of this paper is organized as follows: in Section 2, the discrete phase dynamics at multiscale quantitative levels is proposed. Section 3 presents the design of a stochastic coupling algorithm, including the compensation for frequency drift and communication delay. The stability of this algorithm is verified in Section 4. Simulation and experiment results are discussed in Sections 5 and 6. Finally, conclusions and future research directions are presented in Section 7.

2. Discrete Phase Dynamics

In this section, we generalize the integrating and coupling dynamics. According to the M&S phase model [26], the phase is discretized at a single-scale quantitative level, making it is easier to implement on a WSN node. The single-scale quantitative level is expanded to multiscale quantitative levels for better performance.

2.1. Discretization of a Single-Scale Quantitative Level

In the discrete phase, each node is characterized by a cycle-count timer k (the period is T). Assume that Δt is a short duration of time which denotes the resolution of timer and T can be divided exactly by Δt. The timer k increases by one for each Δt. Thus, in each node, the local timer k can be expressed as a discrete phase, which consists of a series of moments t and:

$$t \in \left\{ k\Delta t \middle| k \in Z, 0 \le k < \frac{T}{\Delta t} \right\} \tag{1}$$

where Z denotes the set of integers, rather than the continuous phase contained in the M&S model. The dynamics of each node are composed of the integrating and coupling dynamics.

2.1.1. Integrating Dynamics

The integrating dynamics are determined by the properties of the node, which does not consider the network coupling. The node's phase would not be influenced by other nodes in integrating dynamics. The integrating dynamics represent the process of increasing the discrete phase k independently in each node as a counter, shown in Equation (2):

$$t^{n+1} - t^n = \Delta t \Rightarrow \left\{ \begin{array}{c} k(t^n) = \frac{T}{\Delta t} - 1 \Rightarrow k(t^{n+1}) = 0 \\ k(t^n) \neq \frac{T}{\Delta t} - 1 \Rightarrow k(t^{n+1}) = k(t^n) + 1 \end{array} \right. \tag{2}$$

As time increases from t^n to t^{n+1} with Δt, the phase k becomes $k + 1$. When the phase reaches its phase threshold $T/\Delta t$, $k(t^{n+1})$ will return to zero immediately. From Equation (2), the period of the phase is T and the update interval is Δt (which denotes the quantization resolution). Here, the quantitative level is defined as the threshold of the timer, as shown in Equation (3):

$$k_{\max} = \frac{T}{\Delta t} - 1 \tag{3}$$

The timer k may be any natural number less than the quantitative level, and T is the threshold time corresponding to the phase threshold.

2.1.2. Coupling Dynamics

The coupling dynamics adjust the phase k of a node coupled with another node in the WSN. For the purposes of the following description, suppose that node j receives the synchronization message of node i. ${}_{i-j}Diff({}_ik(t), {}_jk(t))$ represents the phase difference between i and j at moment t, where the phase in i and j is ${}_ik(t)$ and ${}_jk(t)$, respectively. The phase difference can be calculated from Equation (4), and k can be obtained by processing the synchronization packet. A further explanation is provided in Sections 3.2 and 3.4.

$$ {}_{i-j}Diff({}_ik(t), {}_jk(t)) = \begin{cases} {}_ik(t) - {}_jk(t) + \frac{T}{\Delta t}, when\ {}_ik(t) - {}_jk(t) < -\frac{T}{2\Delta t} \\ {}_ik(t) - {}_jk(t), when\ -\frac{T}{2\Delta t} \le {}_ik(t) - {}_jk(t) \le \frac{T}{2\Delta t} \\ {}_ik(t) - {}_jk(t) - \frac{T}{\Delta t}, when\ \frac{T}{2\Delta t} < {}_ik(t) - {}_jk(t) \end{cases} \tag{4} $$

From Equation (4), ${}_{i-j}Diff$ is an integer and meets $-T/2\Delta t \le {}_{i-j}Diff \le T/2\Delta t$. Considering the discretization, the phase is non-derivable and the adjustment of the phase must be divided exactly by Δt. The adjustment of the discrete phase is determined by a step-increasing function for the discrete phase difference in order to retain the concave property of the phase mapping function [26]. In order to avoid cumbersome floating-point operations, the adjustment is determined according to a step-increasing function $f({}_{i-j}Diff)$ as shown in Equation (5):

$$ f({}_{i-j}Diff) = \begin{cases} 1, & when\quad \frac{T}{2\Delta t} - r > {}_{i-j}Diff > r \\ 0, & when\quad \left|{}_{i-j}Diff\right| \le r \\ -1, & when\quad -r > {}_{i-j}Diff > r - \frac{T}{2\Delta t} \end{cases} \tag{5} $$

In Equation (5), r denotes the refractory period which usually meets $r < 0.4T/\Delta t$ [28]. Here, the refractory period is shown in terms of the phase difference. From Equation (5), the coupling dynamics transform the continuous phase adjustment into a phase state transition, similar to the state transition in a Markov chain. The details are shown in Section 4.1. The conversion conditions are shown in Equation (5), which are key to achieving synchronization. The stability is ensured by the property of Markov chains, which will be explained in Section 4. Finally, the coupling dynamics are presented in Equation (6):

$$ \text{Node } j \text{ coupling with Node } i \Rightarrow \forall j \ne i, {}_jk(t^+) = {}_jk(t) + f({}_{i-j}Diff) \tag{6} $$

Figure 1 shows an example. In this figure, ${}_{i-j}Diff({}_ik(0), {}_jk(0)) = 2$ while the refractory period is 1, and node j is coupling with node i. Thus, $f({}_{i-j}Diff) = 1$. The coupling occurs when the node j reaches its phase coupling ($t = T$). These details will be explained in Section 3.

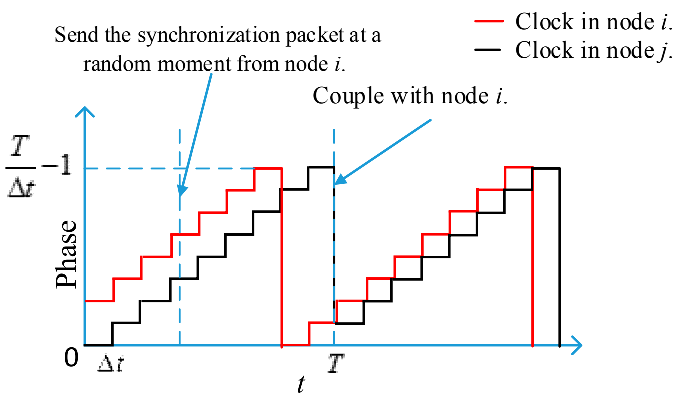


Figure 1. An example of coupling dynamics.

2.2. Discretization of Multiscale Quantitative Levels

Discretization divides the period time into pieces. Quantitative level denotes the amount of pieces. According to [32], a larger quantitative level will lead to a higher synchronization accuracy. However, a large quantitative level will lead to a smaller Δt and a larger number of state elements, which may lead to partial synchronization and slow the convergence. Therefore, multiscale quantization is adopted in this section. Utilizing multiscale quantitative levels, the synchronization process is accelerated by using large quantitative levels, and the accuracy is improved by using small quantitative levels.

Consider the quantization process with different levels. Each level has a different quantization resolution. After quantization, the discrete phase value is converted to a multiscale phase space. In order to decouple each level, we sort them hierarchically by quantitative level and define the threshold time T (T can be modified as needed) in one layer as the quantization resolution in next lower layer, as shown in Equations (7) and (8):

$$k_{l\max}\Delta t_l = \Delta t_{l-1}, T_l = \Delta t_{l-1} \tag{7}$$

$$k_{l\max} = \frac{T_l}{\Delta t_l}(l \in \{0 < l \leq m | l \in Z^+\}) \tag{8}$$

where l denotes the layer index of quantitative level; m is the amount of layers and k_{lmax} represents the quantitative level of layer l. Thus, when layer l is adjusted, the phase in other layers will not be affected unless the threshold is reached. The value of period and quantization resolution in each layer can be selected as needed.

The smallest Δt determines the time resolution of the node, which is usually valued according to the accuracy requirement and hardware limitations. A smaller Δt may get a higher time resolution and accuracy but require a more tedious calculation. Here we use the smallest Δt to form a resolution vector ***RES***, as shown in Equation (9):

$$\begin{gathered} RES_m = 1, RES_l = \prod_{n=l+1}^{m} k_{n\max} \\ \boldsymbol{RES} = \begin{bmatrix} RES_1 \\ \vdots \\ RES_m \end{bmatrix} \end{gathered} \tag{9}$$

Because the time resolution at the highest layer of quantitative level (the mth layer) is the smallest Δt. Thus $RES_m = 1$. In multiscale dynamics, the real phase value can be expressed by the discrete phase value in each layer. Combining the different phase values of m layers forms the state vector ${}_i\boldsymbol{K}(t)$ of the node i, as shown in Equation (10):

$${}_i\boldsymbol{K}(t) = \begin{bmatrix} {}_ik_1(t) \\ {}_ik_2(t) \\ \vdots \\ {}_ik_m(t) \end{bmatrix} \tag{10}$$

where ${}_ik_l(t)$ represents the phase in i at the lth layer at the moment t.

The real normalized phase in node i can be rewritten as ${}_i\boldsymbol{K}$ in Equation (11):

$${}_iPh = \frac{\sum\limits_{l=1}^{m} {}_ik_l \cdot \Delta t_l}{T_{1\max}} = \frac{{}_i\boldsymbol{K}^{\mathrm{T}} \cdot \boldsymbol{RES} \cdot \Delta t_m}{T_{1\max}} \tag{11}$$

In each layer, the integrating and coupling dynamics are the same as the dynamics of the single-scale quantitative level. According to Equation (2), the integrating dynamics at the lth layer are shown in Equation (12).

$$t^{n+1} - t^n = \Delta t_l \Rightarrow \begin{cases} k_l(t^n) = \frac{T_l}{\Delta t_l} - 1 \Rightarrow k_l(t^{n+1}) = 0 \\ k_l(t^n) \neq \frac{T_l}{\Delta t_l} - 1 \Rightarrow k_l(t^{n+1}) = k_l(t^n) + 1 \end{cases} \tag{12}$$

As with the single-scale dynamics, it is necessary to define the multiscale phase differences in a matrix before determining the multiscale coupling dynamics, shown in Equation (14):

$$_{i-j}Diff_l = {}_{i-j}Diff\left({}_ik_l(t), {}_jk_l(t)\right) \tag{13}$$

$$_{i-j}\boldsymbol{Diff}({}_i\boldsymbol{K}, {}_j\boldsymbol{K}) = \begin{bmatrix} {}_{i-j}Diff_1 \\ {}_{i-j}Diff_2 \\ \vdots \\ {}_{i-j}Diff_m \end{bmatrix} \tag{14}$$

where $_{i-j}Diff_l$ denotes the phase difference at the lth layer of quantitative levels between node i and node j.

The step-increasing function can be determined by mapping the phase difference matrix into the adjustment step vector as shown in Equation (15):

$$\boldsymbol{F}\left({}_{i-j}\boldsymbol{Diff}({}_i\boldsymbol{K}, {}_j\boldsymbol{K})\right) = \begin{bmatrix} f({}_{i-j}Diff_1) \\ f({}_{i-j}Diff_2) \\ \vdots \\ f({}_{i-j}Diff_m) \end{bmatrix} \tag{15}$$

where f can be found by Equation (5). Thus, the coupling dynamics at multiscale quantitative levels are shown in Equation (16):

$$\text{Node } j \text{ coupling with Node } i \Rightarrow \forall j \neq i, {}_j\boldsymbol{K}(t^+) = \boldsymbol{K}(t) + \boldsymbol{F}({}_{i-j}\boldsymbol{Diff}) \tag{16}$$

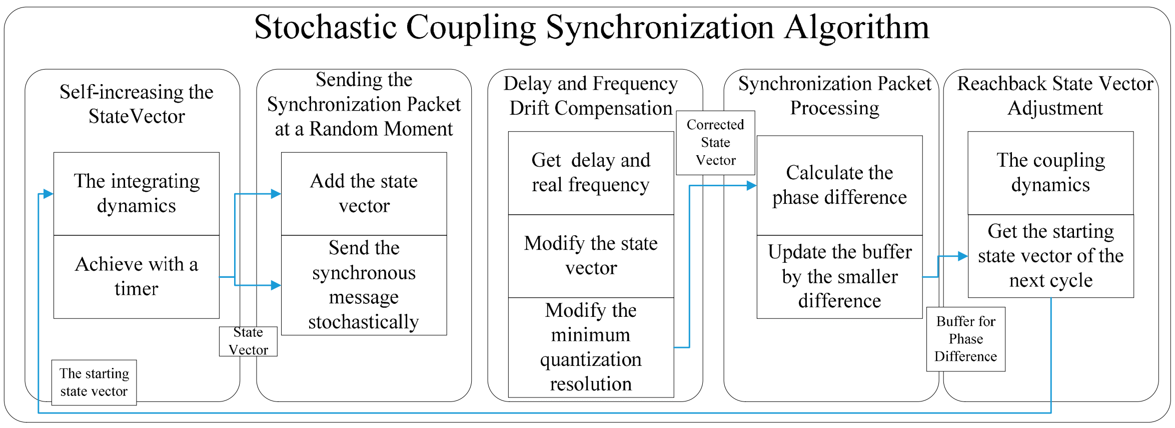

Stochastic Coupling Synchronization Algorithm
Self-increasing the StateVector
The integrating dynamics
Achieve with a timer
The starting state vector
State Vector
Sending the Synchronization Packet at a Random Moment
Add the state vector
Send the synchronous message stochastically
Delay and Frequency Drift Compensation
Get delay and real frequency
Modify the state vector
Modify the minimum quantization resolution
Corrected State Vector
Synchronization Packet Processing
Calculate the phase difference
Update the buffer by the smaller difference
Buffer for Phase Difference
Reachback State Vector Adjustment
The coupling dynamics
Get the starting state vector of the next cycle

Figure 2. The block diagram of the stochastic coupling synchronization algorithm.

3. Stochastic Coupling Synchronization Algorithm

Based on the dynamics presented in Section 2, a stochastic coupling synchronization algorithm is proposed in this section, as shown in Figure 2. It consists of five processing tasks:

1. Self-increasing the state vector.
2. Sending the synchronization packet at a random moment.
3. Delay and frequency drift compensation.
4. Synchronization packet processing.
5. Reachback state vector adjustment.

3.1. Self-Increasing the State Vector

According to the integrating dynamics, the state vector should be updated to meet Equation (12). Unlike the general continuous phase model, the phase growth in each layer is step-like. As described in Section 2, the discrete phase value should increase by one after time Δt_l in the lth layer. Thus, the state vector should be updated at each time point Δt_m, which is easy to achieve with a timer in the MCU.

3.2. Sending the Synchronization Packet at a Random Moment

The algorithm adopts a stochastic coupling method. The triggering synchronization packet includes the regular data format of the wireless network and the state vector information is added at the end of the MAC Payload. It is compatible with existing mainstream network formats (e.g., IEEE802.11 and IEEE802.15.4). Figure 3 illustrates the synchronous message structure of IEEE802.15.4 as an example. The phase information in this figure presents the state vector of the trigger node, and the coupling can be realized in each layer. Unlike the traditional triggering method, the synchronous message sending is carried out stochastically during the synchronization. In addition, a synchronous message can be combined with other network transmission packets to reduce the network load. In addition, we employ the Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) for the idle listening which is compatible with IEEE802.11 or IEEE802.15.4. The Maximum Transmission Unit (MTU) depends on the protocol and the application communication requirements.

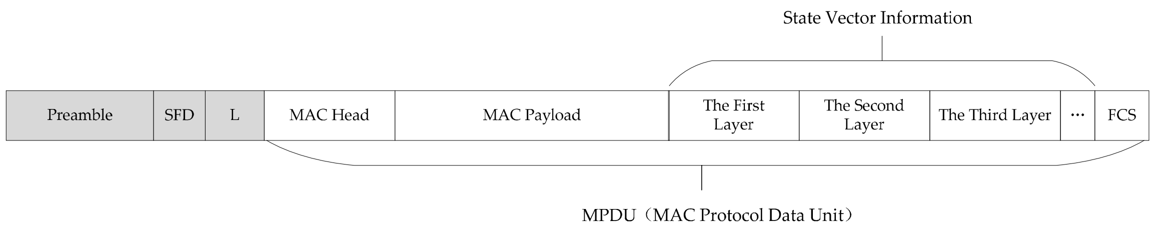


Figure 3. Synchronous message structure.

3.3. Delay and Frequency Drift Compensation

Due to the communication delay and in order to reduce the instability of the algorithm, it is necessary to compensate by optimizing and calibrating the delay. The total communication delay $t_{\rm dly}$ satisfies the relationship in Equation (17):

$$t_{\rm dly} = t_{\rm uc} + t_{\rm c} \approx t_{\rm c} \tag{17}$$

where $t_{\rm uc}$ denotes the uncertain time delay, which includes the sending time, access time, and receiving time. The send time is that of the sender constructing the time message to transmit on the network. The access time is that of the MAC layer delay in accessing the network. Finally, the receive time is the receiving node processing the message and transferring it to the host. $t_{\rm uc}$ can be nearly eliminated with the Start Frame Delimiter (SFD) timestamp technique [28]. In IEEE802.11, chips may not provide the SFD interrupt nor the SFD output pin. Instead, Transmit (TX) start interrupt and Receive (RX) end interrupt can be utilized. Then calculate the timestamp by the TX or RX timing. $t_{\rm c}$ denotes the deterministic delay, which is the difference between the running and response times, which is caused by the interrupted service that is a function of sending and receiving SFD, and can be obtained by calibration. After obtaining the communication delay, it is necessary to convert the delay into the form of a state vector. The conversion method is shown in Equations (18) and (19):

$$k_{\mathrm{dly},l} = (t_{\mathrm{dly}} - \sum_{n=l+1}^{m} k_{\mathrm{dly},n} \Delta t_n) \bmod \Delta t_l \tag{18}$$

$$\boldsymbol{K}_{\mathrm{dly}} = \begin{bmatrix} k_{\mathrm{dly},1} \\ \vdots \\ k_{\mathrm{dly},m} \end{bmatrix} \tag{19}$$

Since the deterministic delay can be calculated in advance, the state vector of the delay can be calculated in advance, and there is no need to calculate it again while the process is running. The modified state vector meets the relationship in Equation (20). Thus, a more accurate state vector $\boldsymbol{K}$ can be obtained:

$$\boldsymbol{K}'(t) = \boldsymbol{K}(t) - \boldsymbol{K}_{\mathrm{dly}} \tag{20}$$

A low-performance RC oscillator is often used as the clock driver in WSNs for cost savings, which introduces a large frequency drift error. To reduce this error and improve the synchronization accuracy, this algorithm employs the correction algorithm presented in [25]. First, the real frequency f_{RC} of the RC oscillator is calibrated and compared with the theoretical frequency f_{ideal}. The adjusted minimum quantization resolution is obtained on the basis of the theoretical minimum quantization resolution Δt_m, shown in Equation (21):

$$\Delta t_1{}' = \frac{\Delta t_1 f_{\mathrm{RC}}}{f_{\mathrm{ideal}}} \tag{21}$$

3.4. Synchronization Packet Processing

In the synchronization process, the node receives the synchronization packet sent by the neighboring node and couples with each layer. During the processing, there is a buffer that only stores the difference of the state vector, which is the smallest in a period of the refractory interval. If we assume that node *j* has received a synchronization packet sent by node *i*, then the process is as follows:

1. Node *j* should record the local state vector ${}_j\boldsymbol{K}$ exactly when receiving the synchronization packet of node *i* and get the corrected state vector by compensation as described in Section 3.3.
2. Analyze the synchronization packet to obtain node *i*'s state vector ${}_i\boldsymbol{K}$.
3. According to Equation (14) calculate the difference ${}_{i-j}\boldsymbol{Diff}(i\boldsymbol{K},j\boldsymbol{K})$. Due to the discretization of phase, when the phase difference equal to 1 in a certain quantitative layer, the real continuous difference value should be less than the time resolution in this layer. So the difference should be reflected in the former layer which has a more precise time resolution. Thus, in order to prevent an oscillatory response between two adjacent layers when the synchronization is almost achieved, it should be equivalently converted to the former layer as shown in Equation (22):

$$\begin{aligned} &{}_{i-j}Diff({}_ik_{l+1}(t), {}_jk_{l+1}(t)) = \pm 1 \\ &\Rightarrow \begin{cases} {}_{i-j}Diff({}_ik_{l+1}(t^+), {}_jk_{l+1}(t^+)) = 0 \\ {}_{i-j}Diff({}_ik_l(t^+), {}_jk_l(t^+)) = {}_{i-j}Diff({}_ik_l(t), {}_jk_l(t)) \pm k_{l\max} \end{cases} \end{aligned} \tag{22}$$

4. If ${}_{i-j}\boldsymbol{Diff}(i\boldsymbol{K},j\boldsymbol{K})$ in Step 3 is out of the refractory interval, update the buffer ${}_{i-j}\boldsymbol{Diff}_{\mathrm{buf}}$ by the smaller difference between the one just obtained and the one stored in the buffer. Thus, after

a period, the buffer only stores the difference of state vector that is the smallest in the period. That is to satisfy Equation (23):

$$\begin{aligned}&\left|{}_{i-j}\boldsymbol{Diff}_{\text{buf}}(t)^{T}\cdot\boldsymbol{RES}\right|>\left|{}_{i-j}\boldsymbol{Diff}({}_{i}\boldsymbol{K},{}_{j}\boldsymbol{K})^{\text{T}}\cdot\boldsymbol{RES}\right|\\&\text{and}\left|{}_{i-j}\boldsymbol{Diff}({}_{i}\boldsymbol{K},{}_{j}\boldsymbol{K})^{\text{T}}\cdot\boldsymbol{RES}\right|\geq r\\&\Rightarrow{}_{i-j}\boldsymbol{Diff}_{\text{buf}}(t^{+})={}_{i-j}\boldsymbol{Diff}({}_{i}\boldsymbol{K},{}_{j}\boldsymbol{K})\\&\left|{}_{i-j}\boldsymbol{Diff}_{\text{buf}}(t)^{T}\cdot\boldsymbol{RES}\right|\leq\left|{}_{i-j}\boldsymbol{Diff}({}_{i}\boldsymbol{K},{}_{j}\boldsymbol{K})^{\text{T}}\cdot\boldsymbol{RES}\right|\\&\text{or}\left|{}_{i-j}\boldsymbol{Diff}({}_{i}\boldsymbol{K},{}_{j}\boldsymbol{K})^{\text{T}}\cdot\boldsymbol{RES}\right|\leq r\\&\Rightarrow{}_{i-j}\boldsymbol{Diff}_{\text{buf}}(t^{+})={}_{i-j}\boldsymbol{Diff}_{\text{buf}}(t)\end{aligned}\tag{23}$$

where $\boldsymbol{RES}$ is the resolution vector in Equation (8) and r is the length of the refractory interval in [28]. The phase difference in the buffer is used to provide input for the reachback state vector adjustment.

3.5. Reachback State Vector Adjustment

As the node's hardware constraints and receiving and adjusting the phase right after receiving the packet may cause hysteresis and instability, the adjustment uses the reachback mechanism of RFA [20]. Under that mechanism, after the longest time threshold, the node obtains the state vector difference in the buffer and calculates $\boldsymbol{F}({}_{i-j}\boldsymbol{Diff}_{\text{buf}})$ as the starting state vector of the next cycle, shown in Equation (24). This can reduce the amount of computing and avoid the impact of disordering reception:

$$k_{m}=k_{m\text{max}}\Rightarrow\boldsymbol{K}=\boldsymbol{F}\left({}_{i-j}\boldsymbol{Diff}_{\text{buf}}\right)\tag{24}$$

4. Stability

As the adjustment in one layer does not affect the other layers, the stability of the target algorithm can be determined by verifying the stability in a single layer. Therefore, this section focuses on proving the stability of the model with a single-scale quantitative level. The stability of the two-node network is demonstrated, and the conclusions are extended to a multi-node network.

4.1. Stability of a Two-Node Network

4.1.1. State Space of the System

As shown in Section 3, all possible values of the discrete phase in node i can be gathered in a set as shown in Equation (25):

$$\begin{aligned}{}_{i}\boldsymbol{X}&=\left\{{}_{i}\phi_{k}=k\Delta\phi,k\in\boldsymbol{D}\right\}\\\boldsymbol{D}&=\left\{n\in\boldsymbol{Z}\middle|0\leq n\leq\tfrac{1}{\Delta\phi}\right\}\end{aligned}\tag{25}$$

To explain this process clearly, suppose T is an even multiple of Δt. Therefore, ${}_{i}\boldsymbol{X}$ is a finite state space of the node itself and each node can be regarded as a finite state machine (FSM), which constitutes a Markov chain.

Similarly, for a two-node network, the absolute value of ${}_{i-j}\mathit{Diff}$ can be selected as the state of the network. All possible values can be gathered in a set as shown in Equation (26):

$$\begin{aligned}{}_{i-j}|Diff|&=\left\{{}_{i-j}|Diff|_{(k)}=k\Delta t,k\in\boldsymbol{D}\right\}\\\boldsymbol{D}&=\left\{k\in Z\middle|0\leq k\leq\tfrac{T}{2\Delta t}\right\}\end{aligned}\tag{26}$$

Hence, the system state can be modeled by a discrete-time Markov chain. Synchronization is achieved when the discrete phase difference between the two nodes decreases to zero.

4.1.2. One-Step Transition Probability Matrix

For nodes *i* and *j*, suppose they satisfy ${}_ik > {}_jk$ and:

$$ik - {}_jk = {}_{i-j}Diff_{({}_ik-{}_jk)} \quad \text{while } {}_{i-j}Diff_{({}_ik-{}_jk)} \leq \frac{T}{2\Delta t} \tag{27}$$

Consider the next adjustment of the phase in a two-node network. From Equation (5), we can see that the adjustment step is $-f({}_{i-j}Diff)$, which always has an opposite sign of the input ${}_{i-j}Diff$. Hence, the adjustment is a negative feedback and always reduces the absolute value of ${}_{i-j}Diff$.

Therefore, if the synchronization has not yet been achieved, the probability of state transition from the adjustment *l* to (*l* + 1) can be found from Equations (28) and (29):

$$P({}_{i-j}\left|Diff\right|^{l+1}_{({}_ik-{}_jk-1)}\Big|{}_{i-j}\left|Diff\right|^{l}_{({}_ik-{}_jk)}) = 1 \tag{28}$$

$$P({}_{i-j}\left|Diff\right|^{l+1}_{({}_ik-{}_jk)}\Big|{}_{i-j}\left|Diff\right|^{l}_{(k)}) = 0 \quad \text{when } k \neq {}_ik - {}_jk - 1,\ {}_{i-j}Diff^{l}_{(k)} \in {}_{i-j}Diff \tag{29}$$

For the two-node system that has a state space shown in Equation (30) and the Markov chain is shown as Figure 4:

$$\begin{bmatrix} 0 & \cdots & n-1 & n & n+1 & \cdots & \frac{T}{2\Delta t} \end{bmatrix}^{\mathrm{T}} \tag{30}$$

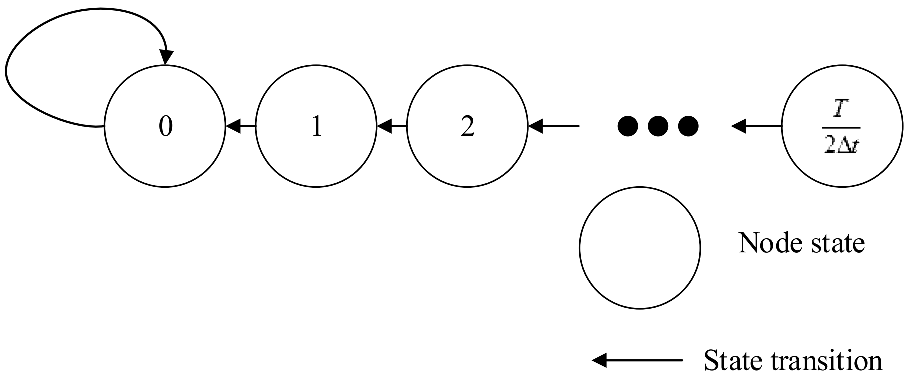


Figure 4. The Markov chain of the two-node system.

The one-step transition probability matrix is constructed as shown in Equation (31):

$$\boldsymbol{P} = \begin{bmatrix} 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{31}$$

4.1.3. Proof of Stability

Suppose the state is n ($n \geq 1$). After one step transition, the state will be $n-1$ according to the Equation (31). And if the state is 0, after one step transition, the state will always be 0. Since the max of state is $T/2\Delta t$ in Equation (30), after $T/2\Delta t$ steps, the state will be 0 constantly and the probability of it will be 1.

From Equation (30), we know that this Markov chain is a finite irreducible chain. The synchronization state is an absorptive state, and has global reachability, producing Equation (32):

$$\boldsymbol{P}^n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}, when\ n \geq \frac{T}{2\Delta t} \tag{32}$$

The two-node system reaches a steady state after $T/2\Delta t$ steps starting from any initial state. Because the steady state is an absorptive state, the algorithm is converged and stable.

4.2. Stability of the Multi-Node Network

The stability of the two-node network can be extended to verify the stability of a multi-node network. In a multi-node network, a specific node can be selected as a reference node, and the corresponding discrete phase differences of the other nodes can be calculated. These phase differences constitute a vector, and all the possible vectors constitute a state space of the multi-node system. The process of the algorithm is transformed into the state transition process of the Markov chain.

According to Equation (31), for any two adjacent nodes, the steady state is an absorbing state and has global reachability while the initial state is a non-absorbing state. As a multi-node network is composed of multiple pairs of two-node networks, when all nodes stochastically match each other, the steady state of the state space also has global reachability [31]. Through deeply connecting and limiting the number of nodes in the network, the composition of the Markov chain is a finite irreducible chain. Therefore, the convergence of the continuous-time Markov process in [31] is also applicable to the discrete-phase Markov chain model of the network. First, the state space in a Markov chain is represented by a set of state vectors $\boldsymbol{S} = \{\boldsymbol{s}_1,\boldsymbol{s}_2, \ldots ,\}$. Each element in S represents a possible state vector of the system. Suppose the steady state in the Markov chain constitutes the set C. The remaining states are non-persistent states, constituting a set N. P represents the Markov chain transition probability matrix, and Equation (33) can be deduced:

$$\boldsymbol{P}^l = (p_{xy}^l) \tag{33}$$

Then:

$$\sum_{y \in \boldsymbol{C} \cup \boldsymbol{N}} p_{xy}^{(l)} = 1 \tag{34}$$

$$\lim_{l \to \infty} \sum_{y \in \boldsymbol{C} \cup \boldsymbol{N}} p_{xy}^{(l)} = 1 \tag{35}$$

$$\lim_{l \to \infty} \sum_{y \in \boldsymbol{N}} p_{xy}^{(l)} + \lim_{l \to \infty} \sum_{y \in \boldsymbol{C}} p_{xy}^{(l)} = 1 \tag{36}$$

For $\forall x,y \in \boldsymbol{N}$, because N is a non-persistent state set:

$$\lim_{l \to \infty} \sum_{y \in \boldsymbol{N}} p_{xy}^{(l)} = 0 \tag{37}$$

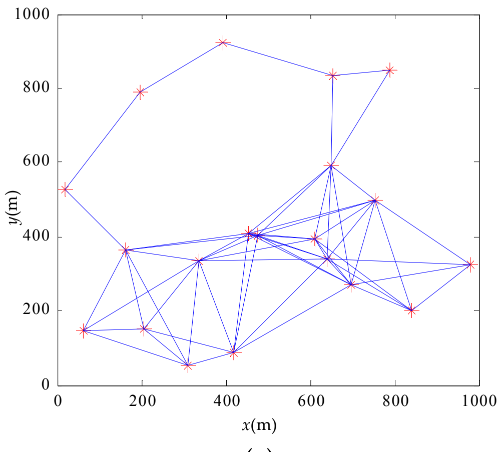
According to the theory of Markov chains, we know that there always exists a value of transfer times n that can realize $\boldsymbol{P}^n > 0$ and $\boldsymbol{P}^{n=1}\boldsymbol{p}^{\mathrm{T}}$, $\boldsymbol{p} \in \boldsymbol{C}$. According to Equation (36), since the steady state is the absorptive state, as the algorithm iterates, other nodes tend to be synchronized and finally reach stability.

5. Simulation Verification

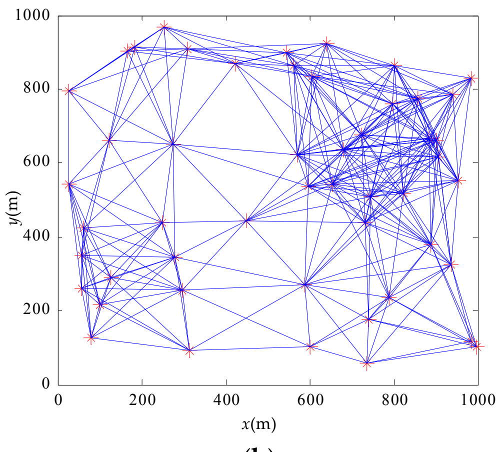
In this section, we verify the stability of the synchronization algorithm proposed in Section 3 using Matlab. We divide the simulations into two groups. One group employs the typical RFA mechanism based on the M&S model and the other one employs the target algorithm to perform synchronization experiments in a non-fully connected mesh network. By recording the phase information of the nodes during the process, the influence of different algorithms on the synchronization can be analyzed.

5.1. Simulation Parameters

In order to enhance the contrast between the algorithms, there are several common simulation parameters in both groups. In the simulation experiment, 30 or 50 nodes are randomly arranged in a 1000 m $\times$ 1000 m area as shown in Figure 5. The communication distance of the node is 350 m, and the simulation time is 300 s with a 4 µs simulation step. The initial phase of the nodes is randomly distributed between 0 and T. The period of the cycle is 1.048576 s. The refractory period of the node is set to 16 µs. The communication delay and frequency drift of the node both influence the synchronization performance [11]. To simulate an extremely poor communication environment in spite of the compensation, we set a 50 ppm frequency drift and 100 µs communication delay. Additionally, a packet loss rate up to 20% is set which is uniformly distributed.



(**a**)



(**b**)

Figure 5. Layout and topology of the network. (**a**) The 20-node network; (**b**) the 50-node network.

The state vector of each node is recorded at intervals of 1 s and the phase difference of any two points in the network can be calculated according to Equation (3). The maximum value of the difference is taken as the synchronization error. In the typical RFA synchronization mechanism, the coupling coefficient is set as 0.5, 0.01, and 0.005. In the multiscale phase dynamics, the number of multiscale layers is set to three, and the quantization levels are 64, 32, and 32. The minimum quantization resolution is 16 µs and the coupling meets Equation (16) while the coupling coefficient is calculated by Equation (15) dynamically.

5.2. Simulation Results

5.2.1. Contrast Test with 20 Nodes

Figure 6 shows the initial phases where 20 nodes are in the non-synchronized state. The synchronization error in the 20-node network is shown in Figure 7. It can be seen that the initial phase of the node set is dispersed in both algorithms. Only the multiscale phase dynamics stimulation achieves phase convergence, and the convergence time is 44 s with an 80 µs error while the standard deviation well is under 26.14 µs. The multiscale phase dynamics algorithm is superior to the RFA in terms of the stability of the algorithm.

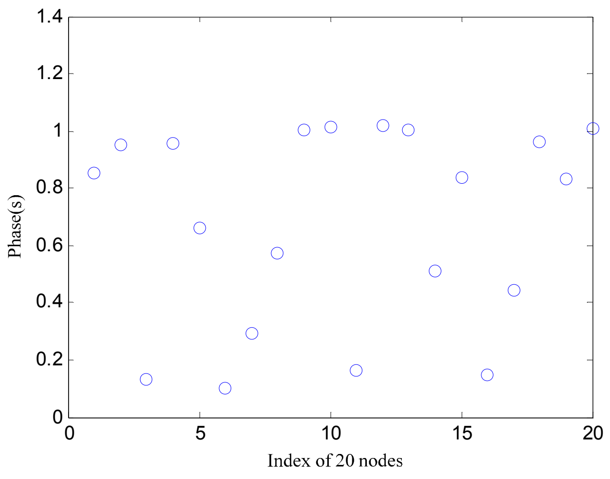


Figure 6. The 20 nodes in non-synchronized states.

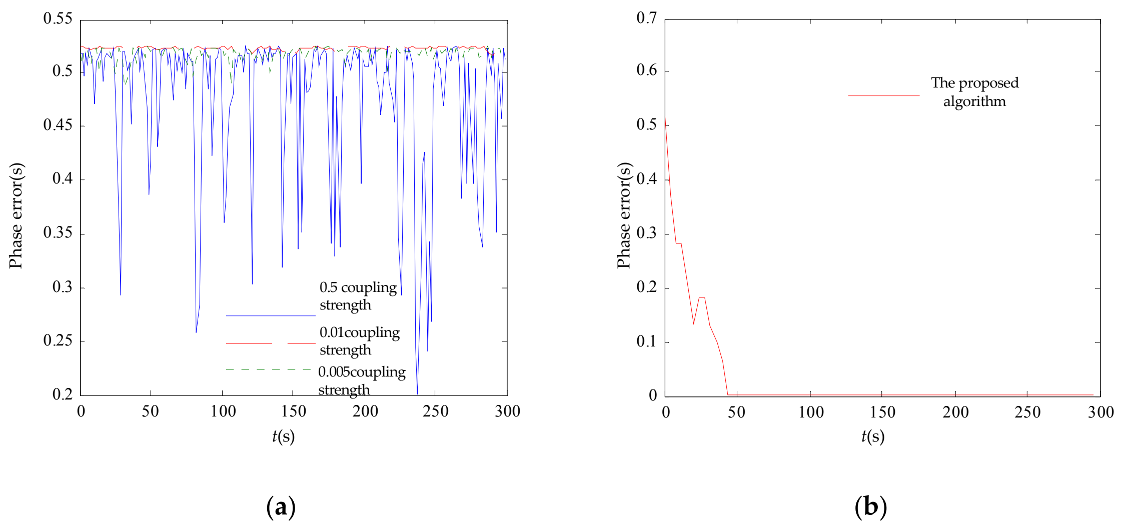


Figure 7. 20-node synchronization error comparison. (**a**) RFA synchronicity error; (**b**) the proposed algorithm synchronicity error.

5.2.2. Contrast Test with 50 Nodes

Figure 8 shows the initial phase where 50 nodes are in the non-synchronized state. The synchronization error in the 50-node network is shown in Figure 9. We can see that the initial phase of the node set is dispersed in both algorithms. The RFA is only able to achieve phase convergence with a coupling strength of 0.01, and the target algorithm achieves phase convergence, while the rest are unstable. The RFA converges in a spread interval of 0.11 s while the target algorithm converges in a spread interval of 0.032 ms with the standard deviation well under 0.00876 ms. These results indicate that the synchronization algorithm based on the proposed mechanism is superior to the traditional RFA in both convergence speed and precision.

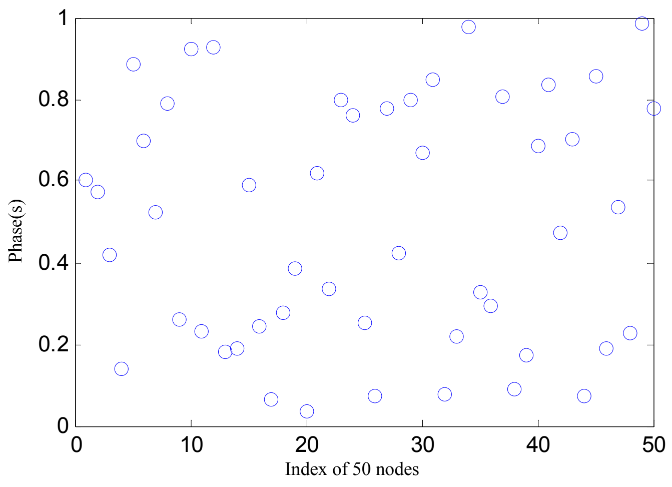


Figure 8. The 50 nodes in non-synchronized states.

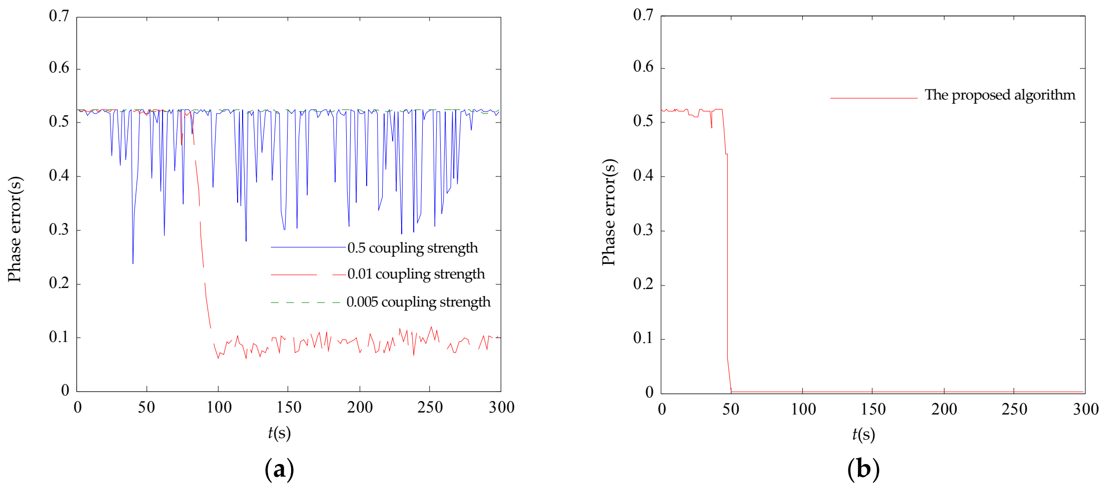


Figure 9. 50-node synchronization error comparison. (**a**) RFA synchronicity error; (**b**) the proposed algorithm synchronicity error.

5.3. Results Analysis and Discussion

5.3.1. Comparison with RFA

The simulation results show that the target algorithm is stable in both networks, while the RFA is stable only in the 50-node network. Additionally, the target algorithm is 50% faster and much more accurate. In terms of synchronization performance, the RFA is difficult to stabilize in the 20-node network because of low network connectivity and needs an appropriate coupling coefficient for convergence in the 50-node network. In contrast, the target algorithm is stable and has a better performance under all conditions. The proposed algorithm is proven to have high stability, high convergence speed, and high synchronization precision compared with RFA. Additionally, the target algorithm has similar advantages as the firefly-inspired algorithm.

5.3.2. Comparison with Traditional Synchronization Protocols

The target algorithm represents a radically different method. Compared with the traditional synchronization protocols, all of the nodes behave equally in our target algorithm. Unlike the traditional synchronization protocols, there are no root nor reference nodes. Our target algorithm is not affected by the network layer. As a result, it is more robust to the single points of failure and the flexible links, which is one of the main attractions. In addition, it is more conducive to network expansion. Nevertheless, comparing with traditional synchronization protocols, there is still a certain gap in terms of accuracy. For instance, as shown in Section 5.2, the target algorithm achieves 32 µs which is better than RFA, but still slightly less than the reported 14 µs accuracy of Flooding Time Synchronization Protocol (FTSP) [9]. We believe that this accuracy can be increased by adjusting the quantization layer and time resolution in each layer and much work remains to be done. Currently, the accuracy of our target algorithm is sufficient for sleep management and time-division multiplexing.

5.3.3. Discussion on Power Consumption

Although the algorithm does not achieve significant improvement in energy consumption, it is still more energy efficient than RFA because of its higher convergence speed. However, as described in Section 2.1.2, each synchronization packet cannot correct the local clock completely but by a little step. Therefore, comparing with the traditional synchronization protocols, more synchronization packets and more time will be needed for reaching the convergence, which means more power consumption. Once the time synchronization is achieved, the synchronization mainly depends on the internal accuracy of the components or clock drifts of the clock system given by the manufacturing process. The traditional synchronization protocols are adaptive. Thus, in this case, they can schedule the synchronization packets on demand. Similarly, if the clock drift is not too large, our target algorithm can reduce the energy consumption by operating at a much lower transmission frequency and combining the synchronization packet with other network transmission packets. The transmission frequency depends on the amount of out-of-phase synchronization packets received during a period of time. So it will be competitive to the traditional synchronization protocols during the maintaining period in terms of the energy consumption. In the future work, we consider the algorithm optimization as follows: When join in the network, each node will initialize its phase by traditional synchronization algorithm, and use our algorithm to tighten the accuracy and maintain synchronicity. It will greatly speed up the synchronization and reduce the energy consumption before convergence.

6. Algorithm Verification on as Hardware Platform

6.1. Hardware and Experiment Design

In order to verify the practicality and validity of the synchronization algorithm, a high-precision synchronization acquisition platform is designed with 30 wireless nodes, an acquisition board, and an industrial computer. The algorithm is implemented on the MCUs (JN5148) in wireless nodes [33]. The phase synchronization emerges from the interaction of the nodes by wireless communication, while the phase value is acquired by a wired National Instruments (NI) synchronization acquisition platform. The unified time reference is provided by the synchronous acquisition to verify the effectiveness of the algorithm.

In this experiment, the integrating dynamics are calculated by the timer on the MAC layer in the RF module, and the synchronization period is 1.0486 s. The minimum quantization resolution is 16 µs. In order to maintain a certain degree of consistency with the simulation, as in the simulation, the number of multiscale layers is three, and the quantization levels are 64, 32, and 32. The refractory period is set to 50 µs. To form a non-fully connected network in the laboratory, it is necessary to restrict each node to have at most eight neighbor nodes with which they can communicate directly, and this limits the network connectivity up to a factor of four. Meanwhile, a packet loss rate up to 20% is set by the software. When a node finishes a synchronization period, its MCU will pull up the level of a GPIO port for 300 ms. The GPIO is sampled by NI's PXIe6358 30-channel digital signal acquisition [34], and the sampling frequency is 100 ksps. For visual observation, the IO port is connected to an LED. The GPIO ports are connected via wires to the digital input channel of the acquisition card while the synchronization is implemented wirelessly. Labview is employed to acquire the real phase value of the

30 nodes on the industrial computer for verification. The block diagram of the acquisition program (the Virtual Instrument block diagram in Labview) is shown as Figure 10.

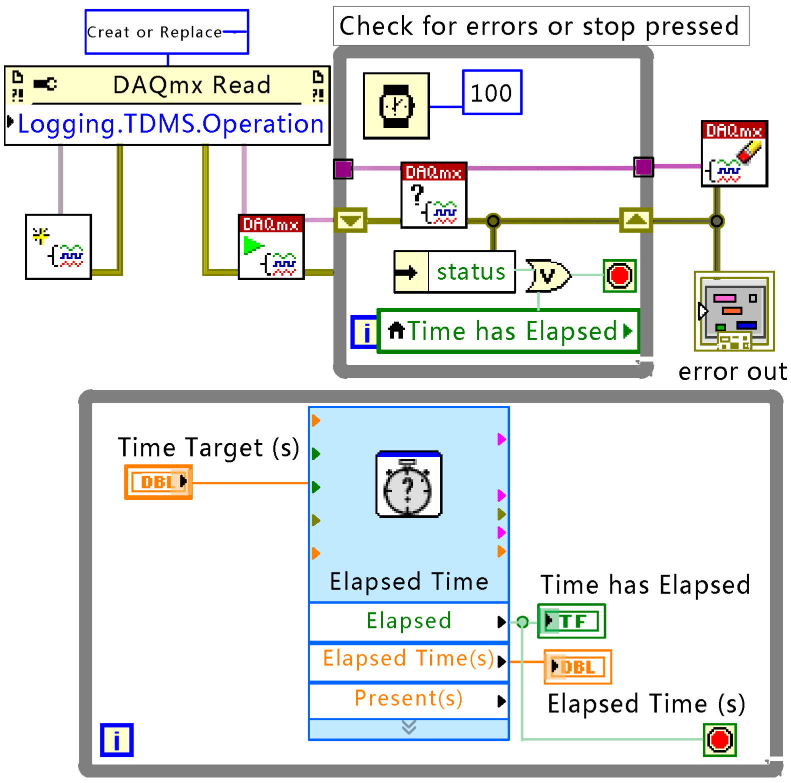
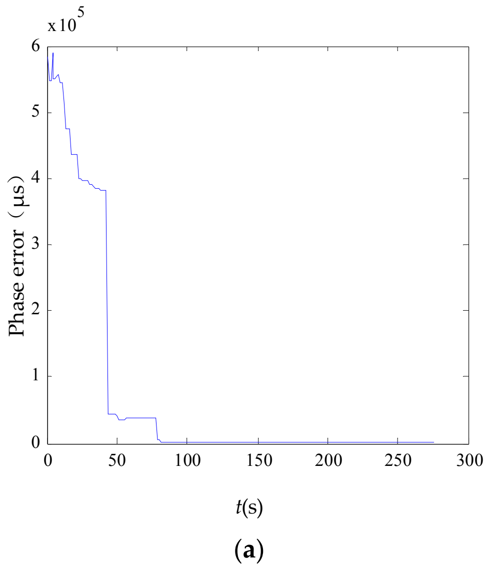


Figure 10. The block diagram of the acquisition program.

6.2. Experimental Results and Discussion

It can be seen in Figure 11 that each node achieves phase synchronization. The maximum error spread of synchronization at each sampling point in time is plotted in Figure 11a. Due to the multiscale phase mechanism, synchronization of the node is divided into three phases. The maximum quantization resolution leads to early rapid decrease of the synchronization error, and the minimum quantization resolution reduces the jitter at later times. Figure 11b shows that the final error spread is in a 50 µs interval with the standard deviation well under 15.42 µs. And the average oscillation period is 1.0485904 s. However, interferences in network may lead to a synchronization error jitter after synchronization, but due to the synchronization algorithm, can still be limited to a certain range. Figure 12 shows the temporal dynamics of 30 nodes in an unsynchronized and synchronized state. In this figure, a vertical bar represents a fire occurrence. Figure 12a shows the chaotic fires in the unsynchronized network. By comparison, Figure 12b has the neat and regular fires in a synchronized state. Figure 13 shows the node synchronization effect. In Figure 13a, the network is in an unsynchronized state. The state at 50 s is shown in Figure 13b, and it can be seen that the network has reached synchronization which requires less than 1600 packets. Therefore, the algorithm achieves synchronization with stability and high performance. The effectiveness of the discrete phase dynamics and algorithm were proven to have a high practical value.



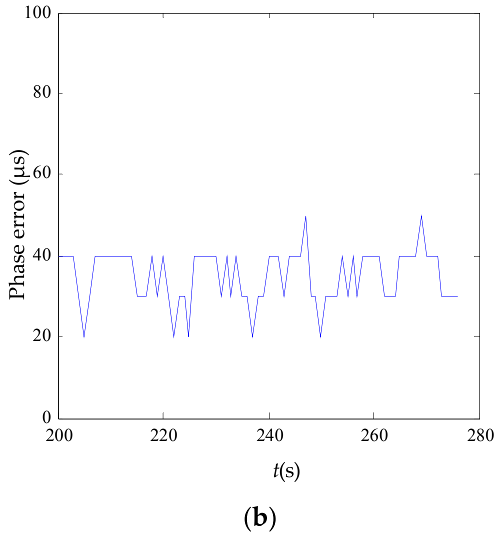
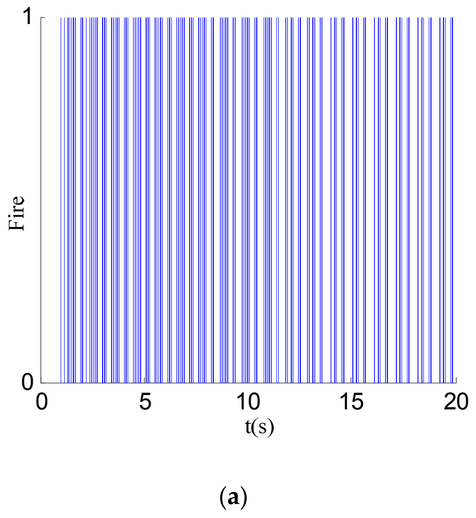


Figure 11. The synchronization error (**a**) during the whole process and (**b**) from 200 to 280 s.



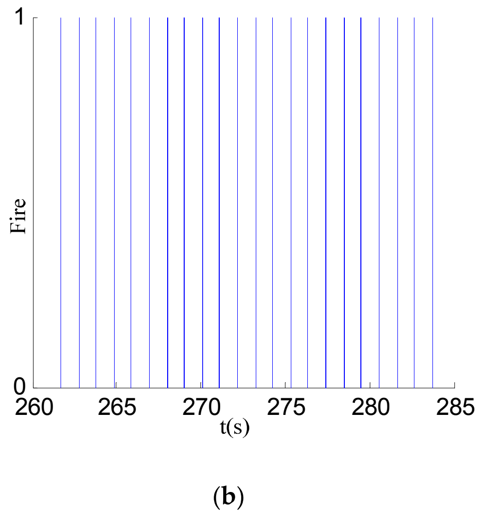


Figure 12. Temporal dynamics of 30 nodes. (**a**) Unsynchronized state; (**b**) synchronized state.

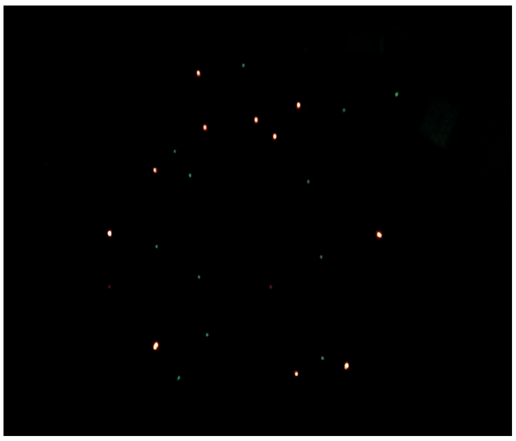
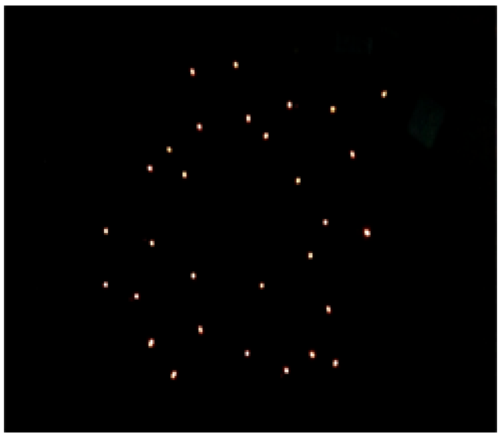

(**a**)

(**b**)

Figure 13. Node synchronization effect. (**a**) Unsynchronized state; (**b**) synchronized state.

7. Conclusions and Extensions

In this paper, discrete phase dynamics at multiscale quantitative levels is proposed. The stochastic coupling algorithm is employed to achieve synchronization in a WSN with a complex topology. The stability of the algorithm is verified in this paper. The simulation results show that the proposed algorithm is more stable in both a 20-node and 50-node network, while RFA is stable only in a 50-node network with a coupling strength of 0.01. Finally, the target algorithm is implemented in a realistic WSN with a complex topology for verification. The performance of the proposed algorithm is only demonstrated preliminarily. It is necessary to conduct further studies to determine the effect of the parameters in the proposed algorithm on the performance and the energy consumption will be taken into account. Because this algorithm is a software-based algorithm, it can and will be transplanted and tested on the ATmega256RFR2 chip for supporting sleep management functions. Nevertheless, the proposed algorithm improves the stability of the firefly synchronization algorithm and reduces the design difficulty of firefly-inspired synchronization algorithms for WSNs with complex topologies.

Acknowledgments: This work was supported in part by the National Natural Science Foundation of China (Grant No. 61202433), the National Key Technology Support Program of China (Grant No. 2013BAC03B02) and the Basic Scientific Research of China (Grant No. JCKY2016602B002).

Author Contributions: Chuangbo Hao and Ping Song designed the research and developed the firefly-inspired synchronization algorithm and the simulation. Cheng Yang and Xiongjun Liu built the verification test platform, designed the software, and analyzed the results.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kadlec, P.; Gabrys, B.; Strandt, S. Data-driven soft sensors in the process industry. *Comput. Chem. Eng.* **2009**, *33*, 795–814. [CrossRef]
2. Budka, M.; Eastwood, M.; Gabrys, B.; Kadlec, P.; Salvador, M.M.; Schwan, S.; Tsakonas, A.; Zliobaite, I. From sensor readings to predictions: On the process of developing practical soft sensors. In Proceedings of the Advances in Intelligent Data Analysis XIII 13th International Symposium, Leuven, Belgium, 30 October–1 November 2014; Volume 8819, pp. 49–60.
3. Averkin, A.N.; Belenki, A.G. Soft computing in wireless sensors networks. In Proceedings of the 5th European Society for Fuzzy Logic and Technology Conference, Ostrava, Czech Republic, 11–14 September 2007; pp. 387–390.
4. Chi, Q.P.; Yan, H.R.; Zhang, C.; Pang, Z.B.; Xu, L.D. A reconfigurable smart sensor interface for industrial wsn in iot environment. *IEEE Trans. Ind. Inf.* **2014**, *10*, 1417–1425.
5. Kumar, S.A.A.; Ovsthus, K.; Kristensen, L.M. An industrial perspective on wireless sensor networks—A survey of requirements, protocols, and challenges. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1391–1412. [CrossRef]
6. Elson, J.; Girod, L.; Estrin, D. Fine-grained network time synchronization using reference broadcasts. In Proceedings of the 2002 Usenix Symposium on Operating Systems Design and Implementation (OSDI'02), Berkeley, CA, USA, 9–11 December 2002; USENIX Association: Berkeley, CA, USA; pp. 147–163.
7. Ganeriwal, S.; Kumar, R.; Srivastava, M.B. Timing-sync protocol for sensor networks. In Proceedings of the First International Conference on Embedded Networked Sensor Systems, Los Angeles, CA, USA, 5–7 November 2003; Association for Computing Machinery: Los Angeles, CA, USA; pp. 138–149.
8. Xu, N.; Zhang, X.; Wang, Q.; Liang, J.; Pan, G.; Zhang, M. An improved flooding time synchronization protocol for industrial wireless networks. In Proceedings of the 2009 International Conference on Embedded Software and Systems, Hangzhou, China, 25–27 May 2009; IEEE Computer Society: Hangzhou, China; pp. 524–529.
9. Djenouri, D.; Bagaa, M. Synchronization protocols and implementation issues in wireless sensor networks: A review. *IEEE Syst. J.* **2016**, *10*, 617–627. [CrossRef]
10. Zhang, H.; Yue, D.; Yin, X.; Hu, S.; Dou, C.X. Finite-time distributed event-triggered consensus control for multi-agent systems. *Inf. Sci.* **2016**, *339*, 132–142. [CrossRef]
11. Simeone, O.; Spagnolini, U.; Bar-Ness, Y.; Strogatz, S.H. Distributed synchronization in wireless networks. *IEEE Signal Process. Mag.* **2008**, *25*, 81–97. [CrossRef]
12. Smith, H.M. Synchronous flashing of fireflies. *Science* **1935**, *82*, 151–152. [CrossRef] [PubMed]

13. Peskin, C.S. *Courant Institute of Mathematical Sciences. Mathematical Aspects of Heart Physiology*; Courant Institute of Mathematical Sciences, New York University: New York, NY, USA, 1975.
14. Izhikevich, E.M. Weakly pulse-coupled oscillators, fm interactions, synchronization, and oscillatory associative memory. *IEEE Trans. Neural Netw.* **1999**, *10*, 508–526. [CrossRef] [PubMed]
15. Okuda, T.; Konishi, K.; Hara, N. Experimental verification of synchronization in pulse-coupled oscillators with a refractory period and frequency distribution. *Chaos* **2011**, *21*. [CrossRef] [PubMed]
16. Klinglmayr, J.; Bettstetter, C. Self-organizing synchronization with inhibitory-coupled oscillators: Convergence and robustness. *ACM Trans. Auton. Adapt. Syst.* **2012**, *7*, 30. [CrossRef]
17. Nunez, F.; Wang, Y.Q.; Doyle, F.J. Synchronization of pulse-coupled oscillators on (strongly) connected graphs. *IEEE Trans. Autom. Control* **2015**, *60*, 1710–1715. [CrossRef]
18. Pruessner, G.; Cheang, S.; Jensen, H.J. Synchronization by small time delays. *Phys. A Stat. Mech. Its Appl.* **2015**, *420*, 8–13. [CrossRef]
19. Núñez, F.; Wang, Y.; Teel, A.R.; Doyle, F.J. Synchronization of pulse-coupled oscillators to a global pacemaker. *Syst. Control Lett.* **2016**, *88*, 75–80. [CrossRef]
20. Werner-Allen, G.; Tewari, G.; Patel, A.; Welsh, M.; Nagpal, R. Firefly-inspired sensor network synchronicity with realistic radio effects. In Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, San Diego, CA, USA, 2–4 November 2005; pp. 142–153.
21. Arellano-Delgado, A.; Cruz-Hernández, C.; López Gutiérrez, R.M.; Posadas-Castillo, C. Outer synchronization of simple firefly discrete models in coupled networks. *Math. Probl. Eng.* **2015**, *2015*, 1–14. [CrossRef]
22. Suedomi, Y.; Tamukoh, H.; Matsuzaka, K.; Tanaka, M.; Morie, T. Parameterized digital hardware design of pulse-coupled phase oscillator networks. *Neurocomputing* **2015**, *165*, 54–62. [CrossRef]
23. Wang, J.Y.; Xu, C.; Feng, J.W.; Chen, M.Z.Q.; Wang, X.F.; Zhao, Y. Synchronization in moving pulse-coupled oscillator networks. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2015**, *62*, 2544–2554. [CrossRef]
24. Mangharam, R.; Rowe, A.; Rajkumar, R. Firefly: A cross-layer platform for real-time embedded wireless networks. *Real-Time Syst.* **2007**, *37*, 183–231. [CrossRef]
25. Rubenstein, M.; Ahler, C.; Hoff, N.; Cabrera, A.; Nagpal, R. Kilobot: A low cost robot with scalable operations designed for collective behaviors. *Robot. Auton. Syst.* **2014**, *62*, 966–975. [CrossRef]
26. Mirollo, R.E.; Strogatz, S.H. Synchronization of pulse-coupled biological oscillators. *SIAM J. Appl. Math.* **1990**, *50*, 1645–1662. [CrossRef]
27. Leidenfrost, R.; Elmenreich, W. Firefly clock synchronization in an 802.15.4 wireless network. *EURASIP J. Embed. Syst.* **2009**, *2009*, 186406–186417. [CrossRef]
28. Cui, L.; Wang, H. Reachback firefly synchronicity with late sensitivity window in wireless sensor networks. In Proceedings of the Ninth International Conference on Hybrid Intelligent Systems, Shenyang, China, 12–14 August 2009; pp. 451–456.
29. Tyrrell, A.; Auer, G.; Bettstetter, C. Emergent slot synchronization in wireless networks. *IEEE Trans. Mob. Comput.* **2010**, *9*, 719–732. [CrossRef]
30. Zhang, Z.S.; Long, K.P.; Wang, J.P.; Dressler, F. On swarm intelligence inspired self-organized networking: Its bionic mechanisms, designing principles and optimization approaches. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 513–537. [CrossRef]
31. LI, L.; Liu, Y.-P.; Yang, H.-Z.; Wang, H. Convergence analysis and accelerating design for distributed consensus time synchronization protocol in wireless sensor networks. *J. Electron. Inf. Technol.* **2010**, *32*, 2045–2051. [CrossRef]
32. Hao, C.; Song, P.; Wu, J.; Yang, C. Multiscale cycle synchronization strategy for wsn based on firefly-inspired algorithm. In Proceedings of the 2015 IEEE International Conference on Information and Automation, Lijiang, China, 8–10 August 2015.
33. Product Brief-JN5148. Available online: http://www.nxp.com/documents/leaflet/JN5148_PB_1v4.pdf (accessed on 15 February 2017).
34. National Instruments, Device Specifications Nation Instruments 6358. Available online: http://www.ni.com/pdf/manuals/374453c.pdf (accessed on 15 February 2017).