

Research Article

Time Synchronization in Wireless Sensor Networks Using Max and Average Consensus Protocol

Zhao Dengchang, An Zhulin, and Xu Yongjun

Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

Correspondence should be addressed to Xu Yongjun; xyj@ict.ac.cn

Received 26 December 2012; Accepted 28 February 2013

Academic Editor: Guangjie Han

Copyright © 2013 Zhao Dengchang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper proposes a novel distributed time synchronization scheme for wireless sensor networks, which uses max consensus to compensate for clock drift and average consensus to compensate for clock offset. The main idea is to achieve a global synchronization just using local information. The proposed protocol has the advantage of being totally distributed, asynchronous, and robust to packet drop and sensor node failure. Finally, the protocol has been implemented in MATLAB. Through several simulations, we can see that this protocol can reduce clock error to ± 10 ticks, adapt to dynamic topology, and be suitable to large-scale applications.

1. Introduction

As in all distributed systems, time synchronization is very important in wireless sensor networks (WSNs) since the design of many protocols and implementation of applications require precise time, for example, forming an energy-efficient radio schedule, conducting in-network processing (data fusion, data suppression, data reduction, etc.), distributing an acoustic beamforming array, performing acoustic ranging (i.e., measuring the time of flight of sound), logging causal events during system debugging, and querying a distributed database.

Time synchronization is a research area with a very long history. Various mechanisms and algorithms have been proposed and extensively used over the past few decades. However, several unique characteristics of WSNs often preclude the use of the existing synchronization techniques in this domain. First, since the amount of energy available to battery-powered sensors is quite limited, time synchronization must be implemented in an energy-efficient way. Second, some messages need to be exchanged for achieving synchronization while limited bandwidth of wireless communication discourages frequent message exchanges among sensor nodes. Third, the small size of a sensor node imposes restrictions on computational power and storage space. Therefore, traditional

synchronization schemes such as network time protocol (NTP) and global positioning system (GPS) are not suitable for WSNs because of complexity and energy issues, cost efficiency, limited size, and so on.

In the context of WSNs, time synchronization refers to the problem of synchronizing clocks across a set of sensor nodes that are connected to one another over a single-hop or multihop wireless networks. To achieve time synchronization in wireless sensor networks, we have to face the following four challenges.

1.1. Nondeterministic Delays. There are many sources of message delivery delays. Kopetz and Ochsenreiter [1] describe the components of message latency, which they call the Reading Error, as being comprised of 4 distinct components plus the local granularity of the nodes clocks. Their work was later expanded by [2] to include transmission and reception time. The most nondeterministic delay is called Access Time, which is incurred in the MAC layer waiting for access to the transmit channel, its orders of magnitude is larger than the synchronization precision required by the network.

1.2. Clock Drift. Manufacturers of crystal oscillators specify a tolerance value in parts per million (PPM) relative

the nominal frequency at 25°C, which determines the maximum amount that the skew rate will deviate from 1. For the nodes used in WSNs, the tolerance value is typically in the order of 5 to 20 PPM. If no drift compensation applied, two synchronized nodes will be out of step soon.

1.3. Robustness. Since sensor networks are often left unattended for long periods of time in possibly hostile environments, synchronization schemes should be robust against link and node failures. Mobile nodes can also disrupt routing schemes, and network partitioning may occur.

1.4. Convergence Speed. Nodes in wireless sensor networks always distribute in large scales, one node may get in touch with another by many hops. This increases the difficulty in reducing the convergence speed in time synchronization algorithm design.

Up to now, many protocols have been designed to address this problem. These protocols all have some basic features in common: a simple connectionless messaging protocol, exchange of clock information among nodes, mitigating the effect of nondeterministic factors in message delivery, and processing utilizing different schemes and algorithms, respectively. They can be classified into two types: centralized synchronization and distributed synchronization.

Centralized synchronization protocol, such as RBS [4], TPSN [2], and FTSP [3], usually has fast convergence speed and little synchronization error. This kind of protocol needs a physical node acting as the whole network's reference clock, so it has to divide the nodes into different roles, for example, client node and beacon node in RBS. If the node with the special role, such as beacon node in RBS, is out of work, the protocol will suffer from big damage. To deal with the WSNs' dynamic topology, centralized synchronization protocol is often designed with complexity logic. Another disadvantage of centralized synchronization protocol is that synchronization error grows with the increase of network hops.

Distributed synchronization protocol, such as TDP [5]/GCS [6]/RFA [7]/ATS [8]/CCS [9], can use local information to achieve the whole network synchronization. This kind of protocol can easily adapt to WSNs' dynamic topology property with lite computation. Currently, the disadvantage of distributed synchronization protocol is that the convergence speed may be a bit slow, relating to the network topology.

This paper describes a new distributed protocol for time synchronization in wireless sensor networks called time synchronization using max and average consensus protocol (TSMA). We adapt a number of techniques to take up the challenges time synchronization has in WSNs. To eliminate the nondeterministic delays, we make use of MAC layer timestamp technique. To compensate for the clock drift, we adapt max consensus protocol, and we use average consensus protocol to compensate for the clock offset. This protocol has the advantages of being computationally light, scalable, asynchronous, robust to node and link failure, and it does not require a master or controlling node.

The rest of the paper is organized as follows. Section 2 summarizes the related work. Section 3 introduces some mathematical tools and definitions that will be instrumental for the proof of convergence of the proposed TSMA algorithm. Section 4 introduces a model for the clock dynamics and formally defines the synchronization objectives, while Section 5 presents the TSMA algorithm in details. This is followed by MATLAB simulations in Section 6. Finally, Section 7 briefly summarizes the results obtained and proposes potential research directions.

2. Related Work

Typical time synchronization algorithms in WSNs include timing-sync protocol for sensor networks (TPSNs) [2], reference broadcast synchronization (RBS) [4], flooding time synchronization protocol (FTSP) [3], time diffusion protocol (TDP) [5], global clock synchronization (GCS) [6], Reach-back Firefly algorithm (RFA) [7], average time synchronization (ATS) [8], and consensus clock synchronization (CCS) [9]. RBS, TPSN, and FTSP are centralized synchronization protocols, while TDP, GCS, RFA, ATS, and CCS are distributed synchronization protocols.

TPSN [2] is designed as a flexible extension of NTP [10] for use in wireless sensor networks, which consists of two phases: a level discovery phase and a synchronization phase. The level discovery phase organizes the whole network into a hierarchical tree topology with the master node at its root, then perform the pair wise synchronization along the branches of the hierarchical structure using the classical sender-receiver synchronization handshake exchange in the synchronization phase. TPSN is a centralized synchronization protocol, utilizes MAC layer timestamping to reduce message delivery nondeterminism and improve synchronization accuracy, its convergence speed increases linearly with the max network hops. This approach suffers from two limitations. The first limitation arises because if the root node or parent node dies, then a new root election or parent-discovery procedure needs to be initiated, thus adding substantial overhead to the code and long periods of network desynchronization. The second limitation is due to the fact that geographically closed nodes might be far in terms of the tree distance, which is directly related to increase the clocks error locally.

RBS [4] aims to provide synchronization amongst a set of client nodes located within the single-hop broadcast range of a beacon node. Compared to the traditional protocols working on an LAN, its main contribution is to directly remove two of the largest sources of nondeterminism involved in message transmission, transmission time, and access time, by exploiting the concept of a time critical path, which is the path of a message that contributes to nondeterministic synchronization errors. This centralized protocol uses least squares linear regression to compensate for the clock drift, its convergence speed also increases linearly with the max network hops. Using the beacon node makes this protocol vulnerable to node failure and node mobility.

FTSP [3] is an ad-hoc, multihop time synchronization protocol for WSNs. It achieves high accuracy by utilizing timestamping of radio messages in low layers of radio stack, completely eliminating access time to the radio channel required in CSMA MAC protocols. Further accuracy is achieved by compensating for clock skews of participating nodes via linear regression. Several mechanisms are used to provide robustness against node and link failures, most notably periodic flooding of synchronization messages throughout the whole network, but this approach still do not completely solve the limitations aforementioned in TPSN.

Nodes employing TDP [5] periodically self-determine to become master/diffusion leader nodes using an election/re-election procedure. Master nodes then engage neighboring nodes in a peer evaluation procedure to isolate problem nodes. Timing information messages are broadcasted from master nodes and then rebroadcasted by the diffused leader nodes for a fixed number of hops, forming a radial tree structure. This approach is fully distributed, but it does not compensate for the clock drift. To the application needing small tolerance time, this algorithm has to increase the synchronization frequency, which greatly decreases the node's battery life.

In GCS [6], nodes take turns to broadcast a synchronization request to their neighbors who each respond with a message containing their local time. The receiving node (at the center of this exchange) averages the received timestamps and broadcasts this value back to its neighbors which adopt this value as their new time. This is repeated by each node in the network until network wide synchronization is achieved. This approach is fully distributed, but it does not compensate for the clock drift. To get higher time accuracy, it has to decrease the time synchronization period.

RFA [7] is inspired by firefly synchronization mechanism. In this algorithm, every node periodically broadcasts a synchronization message, and anytime they hear a message they advance, of a small quantity, the phase of their internal clock that schedules the periodic message broadcasting. Eventually all nodes will advance their phase till they are all synchronized, that is, they fire a message at the same time. This approach, however, does not compensate for clock drift, therefore the firing period needs to be rather small.

ATS [8] uses a cascade of two consensus algorithms to tune compensation parameters and converge nodes to a steady state virtual clock. In the first stage, nodes broadcast local timestamps in order to estimate the clock skew rates relative to each other. Nodes then broadcast their current estimate of the virtual clock skew rate and receiving nodes combine this with their relative skew estimates to adjust their own virtual clock estimate. The same principle is then applied to remove the offset errors. This is a fully distributed protocol; its convergence speed is a bit slow, related to the network topology.

CCS [9] utilizes average consensus algorithm to compensate the clock offset. By the accumulated offset error that they remove during each round of offset compensation nodes can observe how much their own clocks drift away from the consensus time, then they use this information to compensate the clock drift, which can be seen as an enhancement of

TABLE I: Time synchronization features for different protocols.

	Distrib.	Skew comp.	MAC timestamp	Linear skew convergence speed
TPSN [2]	No	No	No	No
FTSP [3]	No	Yes	Yes	Yes
RBS [4]	No	Yes	Yes	Yes
TDP [5]	Yes	No	No	No
GCS [6]	Yes	No	No	No
RFA [7]	Yes	No	No	No
ATS [8]	Yes	Yes	Yes	No
CCS [9]	Yes	Yes	Yes	No
TSMA	Yes	Yes	Yes	Yes

ATS. This approach is also fully distributed and has the disadvantage of slow convergence speed.

The TSMA protocol proposed in this paper is similar to ATS and CCS. It is fully distributed, that is, it does not require any special root, including skew compensation, and exploits MAC-layer timestamping for higher accuracy. Unlike ATS and CCS, the proposed protocol uses max consensus method to compensate the clock drift, which has faster skew convergence speed as the centralized protocol, that is, it increases linearly with the max network hops. The features of all the protocols mentioned in this section are summarized in Table 1.

3. Mathematical Preliminaries

We assume that links in WSNs are symmetric, so the network can be modeled as an undirected graph $G = (V, E)$, where $V = \{v_i \mid i = 1, 2, \dots, n\}$ represents the nodes in the WSNs, and the edge set E represents the available communication links, that is, $(v_i, v_j) \in E$ if node j sends information to node i . We refer to v_i and v_j as the tail and head of the edge (v_i, v_j) , respectively. The orientation of the graph is a choice of heads and tails for each undirected edge. The set of edges of a fix orientation of the graph is denoted by E_0 . Thus, E_0 contains one and only one of the two edges (v_i, v_j) , $(v_j, v_i) \in E$. We use m to present edge number and n the vertex number, for example, $n = |V|$, $m = |E_0|$. The set of neighbors of node i is denoted by $N_i = \{v_j \mid e_{i,j} \in E\}$. The degree of node v_i is the number of its neighbors $|N_i|$ and is denoted by $\deg(v_i)$. The degree matrix is an $n \times n$ matrix defined as $\Delta = \Delta(G) = \{\Delta_{e_{i,j}}\}$, where

$$\Delta_{e_{i,j}} = \begin{cases} \deg(v_i), & i = j \\ 0, & i \neq j. \end{cases} \quad (1)$$

Let A denote the adjacency matrix of G . Then, the Laplacian of graph G is defined by

$$L = \Delta - A. \quad (2)$$

The matrix L has the following features.

- (i) All the row sums of L are zero, and thus $e_0 = (1, 1, \dots, 1)^T \in R^n$ is an eigenvector of L associated with the eigenvalue 0.

- (ii) Fix and orientation of the graph, and let $E_0 = \{e_i \mid i = 1, 2, \dots, m\}$, then $L = CC^T$, where $C = \{c_{pq}\}$ is an $n \times m$ matrix, n is the number of vertex set V , and m is the number of the edge set E_0 ,

$$c_{pq} = \begin{cases} +1, & \text{if } v_p \text{ is the head of edge } e_q; \\ -1, & \text{if } v_p \text{ is the tail of edge } e_q; \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

It is a well-known fact that this property holds regardless of the choice of the orientation of G [11]. Let x_i denote a scalar real value assigned to v_i . Then, $x = (x_1, x_2, \dots, x_n)^T$ denotes the state of the graph G . We define the Laplacian potential of the graph as follows:

$$\Psi_G(x) = \frac{1}{2}x^T Lx. \quad (4)$$

Lemma 1 (Laplacian potential [12]). $\Psi_G(x)$ has the following properties.

- (i) Undirected graph G 's Laplacian potential is positive semidefiniteness, and

$$x^T Lx = \sum_{(v_i, v_j) \in E_0} (x_i - x_j)^2. \quad (5)$$

- (ii) If G is a connected graph, $\Psi_G(x) = 0$ if and only if $x_i = x_j$, for all i, j .

Lemma 2 (connectivity and graph Laplacian [11]). Assume graph G has c connected components, then

$$\text{rank}(L) = n - c. \quad (6)$$

Particularly, for a connected graph with $c = 1$, $\text{rank}(L) = n - 1$.

Lemma 3. G is an undirected and connected graph, L is G 's Laplacian matrix, then there must be a zero in L 's eigenvalues, and all the other eigenvalues of G are positive and real.

Proof. G is an undirected graph, then L is a real symmetric matrix. After diagonalization, the rank of matrix L stays the same. G is a connected graph, then based on Lemma 2, the rank of matrix L is $n - 1$, then there must be a zero in L 's eigenvalues, and all the other eigenvalues of G are nonzero. L is a real symmetric matrix, so L 's eigenvalues are real. Based on Lemma 1, L is a positive semidefinite matrix, then L 's eigenvalues are nonnegative. In conclusion, there must be a zero in L 's eigenvalues, and all the other eigenvalues of G are positive and real. \square

Definition 4 (consensus). Let the value of all nodes x be the solution of the following differential equation:

$$\dot{x} = f(x), \quad x(0) \in R^n. \quad (7)$$

In addition, let $X : R^n \rightarrow R$ be a multi-input single-output operation on $x = (x_1, x_2, \dots, x_n)^T$ that generates

a decision value $y = X(x)$. We say all the nodes of the graph have reached consensus w.r.t. X in finite time $T > 0$ if and only if all the nodes agree and $x_i(T) = X(x(0))$, for all $i \in I$. Some of the common examples of the operation X are given as follows:

- (i) average consensus,

$$X(x) = \text{Ave}(x) = \frac{1}{n} \sum_{i=1}^n x_i; \quad (8)$$

- (ii) max consensus,

$$X(x) = \text{Max}(x) = \max(x_1, x_2, \dots, x_n); \quad (9)$$

- (iii) min consensus,

$$X(x) = \text{Min}(x) = \min(x_1, x_2, \dots, x_n). \quad (10)$$

Theorem 5. G is an undirected and connected graph if every node in G runs the following distributed protocol:

$$\dot{u}_i(t) = \sum_{j \in N_i} (x_j(t) - x_i(t)) + T_0, \quad (11)$$

where T_0 is a constant, then the vector of the value of the nodes x is the solution of the following ordinary differential equations:

$$\dot{x} = -Lx + (T_0, T_0, \dots, T_0)^T, \quad x(0) \in R^n. \quad (12)$$

In addition, all the nodes of the graph globally asymptotically reach an average consensus, and then every node will have the same status,

$$x_i(t) = x_j(t) = tT_0 + \text{Ave}(x(0)), \quad \forall i, j, i \neq j. \quad (13)$$

Proof. Equation (12) corresponds to a typical linear time invariant system; its complete response can be solved by calculating zero-status response and zero-input response, respectively.

- (a) Solve the zero-status response $x_s(t)$, where

$$\dot{x}_s = -Lx_s + (T_0, T_0, \dots, T_0)^T, \quad x_s(0) = (0, 0, \dots, 0)^T. \quad (14)$$

Based on Laplacian matrix L 's features, the particular solution of $x_s(t)$ is $x_p(t) = T_0(t, t, \dots, t)^T$. $-L$ is a real symmetric matrix, so $-L$ can do diagonalization, let P be the corresponding invertible matrix. Based on the theory of linear algebra, the homogeneous solution of $x_s(t)$ is

$$x_h(t) = c_1 \alpha_1 e^{\lambda_1 t} + c_2 \alpha_2 e^{\lambda_2 t} + \dots + c_n \alpha_n e^{\lambda_n t}, \quad (15)$$

where c_i ($1 \leq i \leq n$) is undetermined coefficient, λ_i ($1 \leq i \leq n$) is matrix L 's i th eigenvalue, α_i ($1 \leq i \leq n$) is P 's i th column, which is the corresponding eigenvector of eigenvalue λ_i , so the zero-status's complete response is

$$x_s(t) = x_p(t) + x_h(t) = T_0(t, t, \dots, t)^T + c_1 \alpha_1 e^{\lambda_1 t} + c_2 \alpha_2 e^{\lambda_2 t} + \dots + c_n \alpha_n e^{\lambda_n t}. \quad (16)$$

We know that $x(0) = (0, 0, \dots, 0)^T$, so $c_1\alpha_1 + c_2\alpha_2 + \dots + c_n\alpha_n = (0, 0, \dots, 0)^T$, in addition, $\alpha_1, \alpha_2, \dots, \alpha_n$ is linear independence, then $c_1 = c_2 = \dots = c_n = 0$, the solution of zero-status response is $x_s(t) = T_0(t, t, \dots, t)^T$.

(b) Solve the zero-input response $x_s(t)$, where

$$\dot{x}_0 = -Lx_0, \quad x_0(0) = x(0). \quad (17)$$

Observe that all the eigenvalues of $-L$ are negative, except for the single zero eigenvalue. Thus, any solution of the system asymptotically converges to a point x_0^* in the eigenspace associated with $\lambda_1 = 0$. This implies that an average consensus is globally asymptotically reached by all the nodes. Let x_0^* be one equilibrium of the system $\dot{x}_0 = -Lx_0$. Then, $-Lx_0^* = 0$, and thus x^* is the eigenvector of the Laplacian L associated with the zero eigenvalue $\lambda_1 = 0$. On the other hand, we have $\psi_G(x_0^*) = (1/2)(x_0^*)^T Lx_0^* = 0$, and from connectivity of G , it follows that $x_{0,i}^* = x_{0,j}^* = a$, for all i, j , that is, $x_0^* = (a, a, \dots, a)^T$, $a \in \mathbb{R}$. Notice that $\sum_{i=1}^n x_{0,i} = 0$. Thus, $\text{Ave}(x_0)$ is an invariant quantity. This implies $\text{Ave}(x_0^*) = \text{Ave}(x(0))$. But $\text{Ave}(x_0^*) = a$, therefore $a = \text{Ave}(x(0))$, then the solution of zero-input response is $x_0^* = (a, a, \dots, a)^T$, where $a = \text{Ave}(x(0))$.

Above all, when zero-input response reaches the equilibrium status, the system's full response is

$$\begin{aligned} x(t) &= x_s(t) + x_0(t) \\ &= (tT_0 + \text{Ave}(x(0)), tT_0 + \text{Ave}(x(0)), \dots, tT_0 \\ &\quad + \text{Ave}(x(0)))^T. \end{aligned} \quad (18)$$

In other words, every node in the network finally has the same status. \square

4. Clock Model

Clock synchronization is of significant importance in WSNs. Before delving into the details of synchronizing clocks, we first define some terminologies used in this paper. $c(t)$ denotes a reference clock, where t is the real time, that is, coordinated universal time (UTC). Every sensor node maintains its own local clock, which is a monotonically nondecreasing function of t . This local clock is an ensemble of hardware and software components, essentially a timer that counts the oscillations of a quartz crystal running a particular frequency. In general, the timer is programmed to generate an interrupt, which is called a clock tick. At each clock tick, the interrupt procedure increments the clock value stored in memory.

For any two nodes' local clocks $c_i(t)$ and $c_k(t)$, the clock $c_i(t)$ is considered correct at time t if $c_i(t) = c(t)$, or the clock $c_i(t)$ is considered accurate at time t if $dc_i(t)/dt = dc(t)/dt$, and two clocks $c_i(t)$ and $c_k(t)$ are synchronized at time t if $c_i(t) = c_k(t)$. The aforementioned definitions show that the two synchronized clocks are not always correct or accurate; time synchronization is not related to time correctness and accuracy. As for most applications in WSNs, it is sufficient to achieve clock synchronization. Since the oscillator frequency

is time varying due to ambient conditions such as temperature changes, variations of electric supply voltage, and air pressure, clock $c_i(t)$ of node i can be modeled as

$$c_i(t) = k_i t + b_i, \quad (19)$$

where b_i is the clock offset, the difference between the time reported by clock $c_i(t)$ and the real time at the initial instant t_0 (i.e., $b_i = c_i(t_0) - c(t_0)$), and k_i is defined as the skew of the clock $c_i(t)$. To achieve perfect synchronization in a sensor network, all nodes $i = 1, 2, \dots, N$ must precisely compensate for their clock parameters k_i and b_i so that all clocks have zero offset error and all tick at the same rate.

5. TSMA Method

In this section, we present our time synchronization algorithm, that is, TSMA. Instead of trying to synchronize to an external reference like absolute time or UTC, the TSMA method aims to achieve an internal consensus within the network on what time is, and how fast it travels. In every synchronization round, the TSMA algorithm updates the compensation parameters for each node and over time the network clocks converge to a consensus,

$$\lim_{t \rightarrow \infty} c'_i(t) = c_c(t), \quad (20)$$

where $c_c(t)$ is the consensus clock.

The consensus clock is not a physical clock, it is a new virtual clock, that is, generated from the network of nodes running the TSMA algorithm. The consensus clock has its own skew rate and offset relative to the absolute time.

By expanding the clock functions from both sides of (20), we can see the compensation parameters that all nodes must obtain in order to synchronize to the consensus clock,

$$\lim_{t \rightarrow \infty} (k'_i c_i(t) + b'_i) = k_c t + b_c, \quad (21)$$

$$\lim_{t \rightarrow \infty} (k'_i k_i t + k'_i b_i + b'_i) = k_c t + b_c.$$

Then, we have

$$\lim_{t \rightarrow \infty} k'_i = \frac{k_c}{k_i}, \quad (22)$$

$$\lim_{t \rightarrow \infty} b'_i = b_c - k'_i b_i.$$

To achieve these compensation parameters, nodes repeat the TSMA algorithm in synchronization rounds which consist of two main tasks; skew compensation and offset compensation. In skew compensation, the nodes iteratively select the max skew rate estimation from its neighbors in order to improve their skew compensation parameter. In the offset compensation phase, nodes average their neighbors' clock readings to synchronize nodes to a common time.

5.1. Skew Compensation. The goal of skew compensation is to ensure all compensated clocks in the network tick at the same rate, that is,

$$\lim_{t \rightarrow \infty} k'_i k_i = k_c, \quad \forall i. \quad (23)$$

We use max consensus protocol to compensate for clock skew. For perfect skew compensation, the skew rate of the consensus clock is given by

$$k_c = \max(k_1, k_2, \dots, k_n). \quad (24)$$

Nodes execute the following algorithm to converge their clock skew to the consensus value.

Step 1. Before running the synchronization algorithm, nodes set their skew estimation k'_i to 1, and set their respect life time $life_i$ to 0.

Step 2. At the beginning of each synchronization round, every node increases their own life time $life_i$ by 1. If node i 's life time $life_i > 3$, then node i broadcasts a synchronization packet containing its own local clock value τ_i and estimated skew k'_i using a CSMA/CA MAC scheme once per synchronization round. Like other synchronization techniques, MAC layer timestamping is required to ensure that clock values are transmitted instantly.

Step 3. If node i receives a synchronization packet from node j at time t , then this node should immediately record its own local time $\tau_i(t)$ and node j 's information $(\tau_j(t), k'_j(t))$. If node i has not stored a history local time pair before, then create the local time pair $(\tau_i(\text{old}), \tau_j(\text{old}))$ and assign it with

$$(\tau_i(\text{old}), \tau_j(\text{old})) = (\tau_i(t), \tau_j(t)). \quad (25)$$

If the history local time pair has already existed, and satisfies $k'_i(t) * (\tau_j(t) - \tau_j(\text{old})) > k'_j(t) * (\tau_i(t) - \tau_i(\text{old}))$, we will update this node's estimated skew k'_i and the history local time pair $(\tau_i(\text{old}), \tau_j(\text{old}))$ using

$$k'_i(t) = k'_j(t) * \frac{\tau_i(t) - \tau_i(\text{old})}{\tau_j(t) - \tau_j(\text{old})}, \quad (26)$$

$$(\tau_i(\text{old}), \tau_j(\text{old})) = (\tau_i(t), \tau_j(t)).$$

5.2. Offset Compensation. After the skew compensation algorithm is applied, all local estimators will eventually have the same drift, that is, they will run at the same speed. At this point, it is only necessary to compensate for possible offset errors. To achieve this, each node aims to accurately estimate the instantaneous average time of all its neighbors' clock in the network and set their clocks to this time. From Theorem 5, we can see that every node in the network will finally reach the same status,

$$\bar{b}(t) = \frac{1}{|N_i|} \sum_{j \in N_i} b_j(t). \quad (27)$$

In reality, however, it is not practically possible for nodes to calculate the instantaneous global average time of all its neighbors' clock in the network. Since sensor networks may consist of a large number of nodes, and one node may have lots of neighbors. It is impossible for nodes to exchange clock values instantaneously, and nodes may suffer from packet loss and other network dynamics.

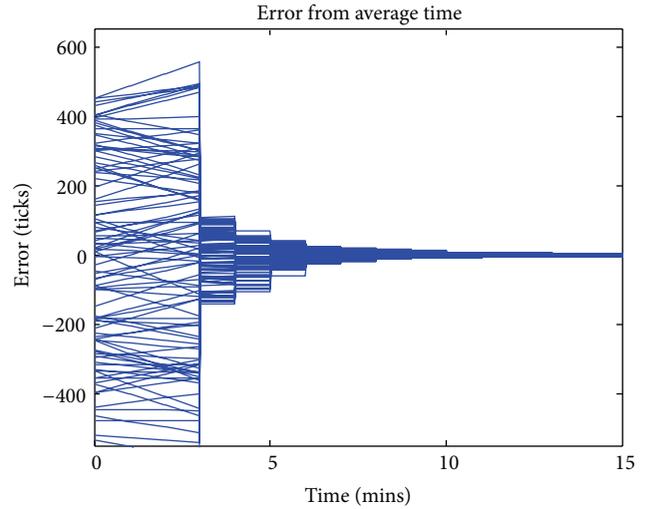


FIGURE 1: Clock error of 100 simulated nodes running the TSMA protocol in a 10 * 10 network with synchronization period of 1 min and transmission range of 2.

Instead, the TSMA algorithm converges network clocks to an approximation of $\bar{b}(t)$ using the locally available information within the single-hop transmission range of each node. Nodes execute the following algorithm to converge their clock offset to the consensus value.

Step 1. Before running the synchronization algorithm, nodes set their offset estimation b'_i to their current local time and set their respect life time $life_i$ to 0.

Step 2. At the beginning of each synchronization round, nodes reset their confidence parameter $Conf_i$ to 1 and increase their life time $life_i$ by 1. If node i 's life time $life_i > 3$, then node i broadcasts a synchronization packet containing its estimated offset b'_i and confidence parameter $Conf_i$ using a CSMA/CA MAC scheme once per synchronization round. MAC layer timestamping is required to ensure that clock values are transmitted instantly.

Step 3. For each broadcast, all other nodes within receiving range update their own clocks by combining this value with their own clocks using the confidence weighted running average equation and then increase their confidence parameter $Conf_i$ by 1,

$$b'_i(t) = \begin{cases} b'_j(t), & \text{if } life_i = 1 \\ \frac{Conf_i(t) * b'_i(t) + Conf_j(t) * b'_j(t)}{Conf_i(t) + Conf_j(t)}, & \text{if } life_i > 1, \end{cases} \quad (28)$$

$$Conf_i(t) = Conf_i(t) + 1.$$

The confidence parameters give weighting to the clock values from nodes that have already received synchronization packets in the current round. These nodes are generally closer to a local consensus than those who have not received

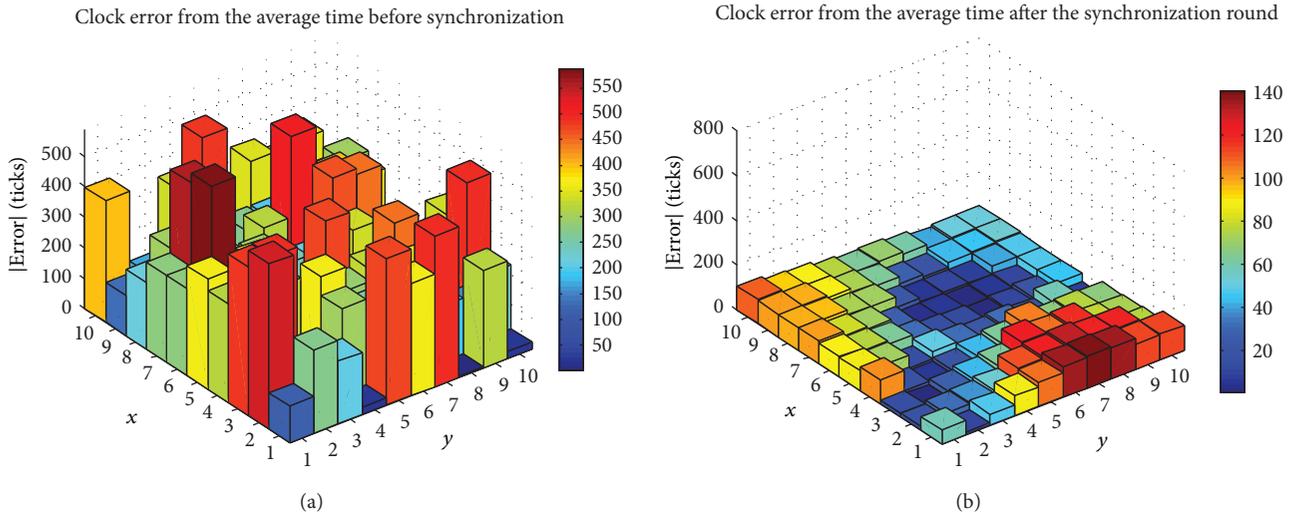


FIGURE 2: Geographic distribution of synchronization error of a $10 * 10$ sensor network for transmission range of 2 (a) before and (b) after one round of offset compensation.

any sync packets, and this was found to increase the rate of convergence.

6. Simulation

In this paper, we use max consensus to compensate for clock skew and use average consensus to compensate for clock offset. To show the correctness and robustness of the TMSA algorithm, we make a simulation in MATLAB; the following is the simulation environment: there are 100 WSN nodes in the $10 * 10$ grid, one grid one node, and we place the node in the center of the grid, the base value of nodes' oscillator is 32.768 KHz, the skew rates are assigned randomly from a normal distribution with mean = 1 and standard deviation = 20 PPM, nodes are given an initial error distribution between 0 and 1000 ticks.

6.1. Offset Compensation. In this simulation, we set the time synchronization period to 1 minute. Nodes broadcast one sync packet per round in random order with a transmission range of 2, which means if the distance between two nodes is less than 2, then they can communicate with each other. Figure 1 shows the clock errors from average time measured in clock cycles. In the first three synchronization periods, nodes' life time is less than 3, so they do not transmit any synchronization packet to their neighbors, and then in 3 minutes, nodes start to compensate for the clock skew and offset. After seven rounds compensation, 100 nodes in the network achieve the synchronization state, the synchronization error between any two nodes is included between ± 10 ticks.

The geographic distribution of clock errors in the network is shown in Figure 2. It can be seen that before Figure 2(a), the first round of offset compensation, the clock error relative to the average time can reach up to 550 ticks, but after Figure 2(b), the first round offset compensation, the clock error reduces to 140 ticks. Also we can see that the clock

error after compensation transfer smoothly in geographic distribution, the local error between any neighboring nodes is no larger than 50 ticks.

A histogram of the clock errors is shown in Figure 3. It can be seen that before Figure 3(a), the first round of offset compensation, the standard deviation of clock error reaches up to 286.32, but after Figure 3(b), the first round of offset compensation, this value reduces to 70.90. From Figures 2 and 3, we can see that after the first round of offset compensation, the clock errors among nodes are significantly reduced.

Figure 4 shows the convergence of the skew rates of 100 nodes in this simulation. From Figure 4, we can see that after two synchronization periods all the 100 nodes' skew rates converge on the maximum clock skew.

6.2. Transmission Range. Figure 5 shows the transmission range's effect to algorithm performance. We set the transmission range to 1, 2, and 3, respectively, then compare their convergence speed with each other. From Figure 5, we can see that synchronization accuracy and convergence speed are found to improve as the transmission range of the nodes increased, since nodes could receive synchronization packets from a greater proportion of the network population. On the other side, to increase the transmission range, we also increase the energy consumption of nodes, so in the practical application, we should make a tradeoff between energy consumption and convergence speed.

6.3. Dynamic Topology. The simulation, shown in Figure 6, is intended to study the robustness properties of the TMSA protocol subject to node failure and node replacement, as well as the performance in terms of convergence speed and steady state synchronization error. The synchronization period is set to 1 minute which is sufficiently large to exhibit the effects of different clock speeds. The simulation was run for about

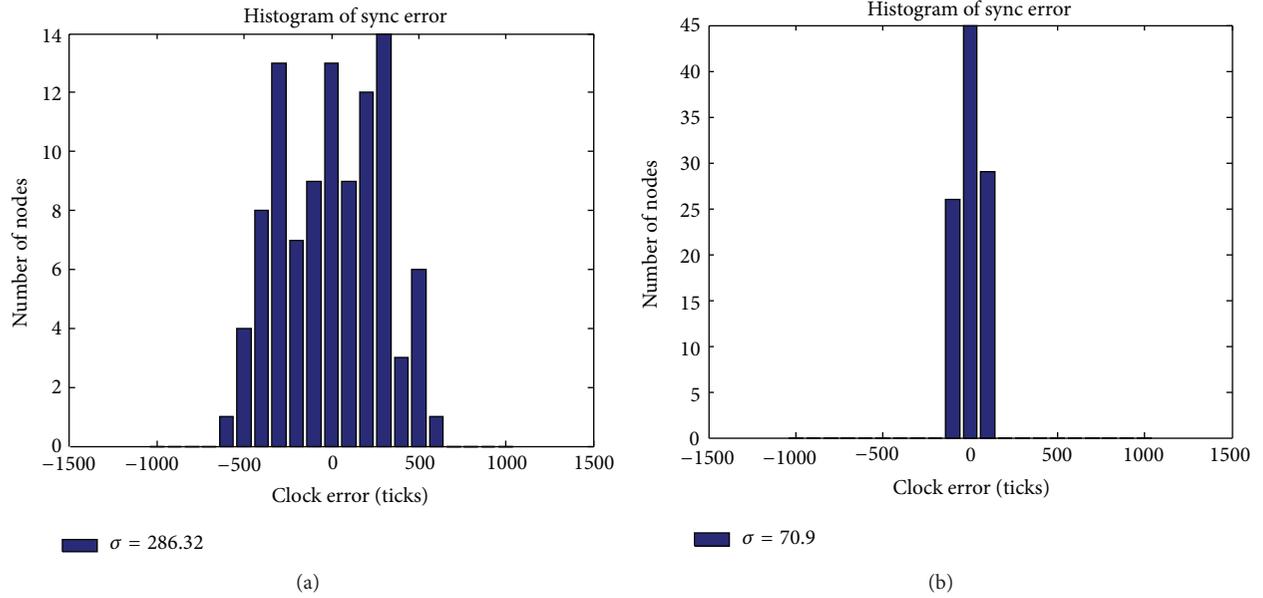


FIGURE 3: Histogram of the synchronization errors from the instantaneous global average time of the sensor network (a) before and (b) after one round of offset compensation.

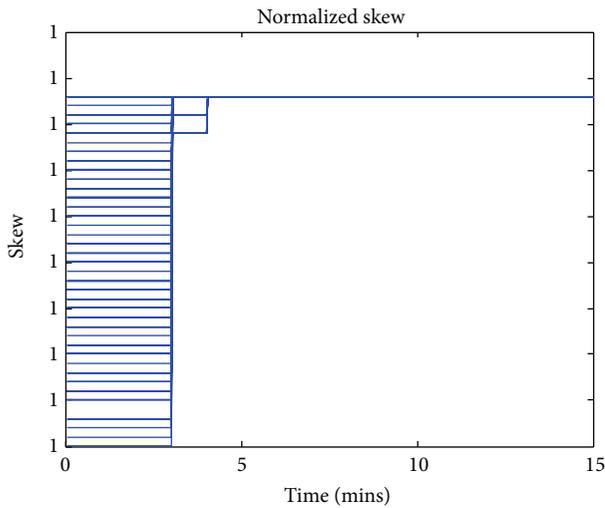


FIGURE 4: Convergence of the skew rate of 100 simulated nodes running the TSMA protocol in a $10 * 10$ network with synchronization period of 1 min and transmission range of 2.

50 minutes and presents 4 different regions of operation indicated by the letters A, B, C, and D which model potential node failure or the addition of the new nodes. In Region A, all nodes are turned on simultaneously with random initial conditions of their local clocks. After 2 synchronization periods, the synchronization error between any two nodes is included between ± 10 ticks, that is, the maximum error is smaller than 20 ticks. At the beginning of Region B, about 20% of the nodes chosen random in the grid are switched off and then switched on at different random times. Once a node is switched on, it starts updating its estimated time

using TSMA protocol but does not transmit any message for the first three synchronization periods to avoid injecting large disturbances into the already synchronized network, and then it starts to transmitting and receiving messages equally. The plot in Figure 6 clearly shows that the nodes get synchronized as soon as they are turned on without perturbing the overall network behavior. At the beginning of Region C, about 20% of the nodes turned off their radio, that is, they stop updating their parameters k' , b' , then we turn on the 20% nodes' radio at about 35 minutes. In this procedure, we can see that all nodes keep the synchronized state, the reason is that their skew rates have already converged to the maximum skew rate in the network, these 20% nodes still synchronize with others. At the beginning of Region D, we replace the node in the top left corner with one node having 40 PPM drift, which is the max skew rate in the network now, and then all the other nodes will compensate themselves to this skew value. From Figure 6, we can see that three synchronization period after the new node joining the network, it starts to transmitting messages to its neighbors, after a short transient the nodes quickly synchronize again.

7. Conclusions

This paper presented a new synchronization algorithm for WSN, the time synchronization using consensus protocol, which is based on consensus algorithms whose main idea is to average local information to achieve a global agreement on a specific quantity of interest. The proposed algorithm is fully distributed, asynchronous, includes skew compensation, and is computationally light. Moreover, it is robust to dynamic network topologies due, for example, to node failure or replacement. Finally, a set of simulations are presented to show the good performance of the proposed protocol.

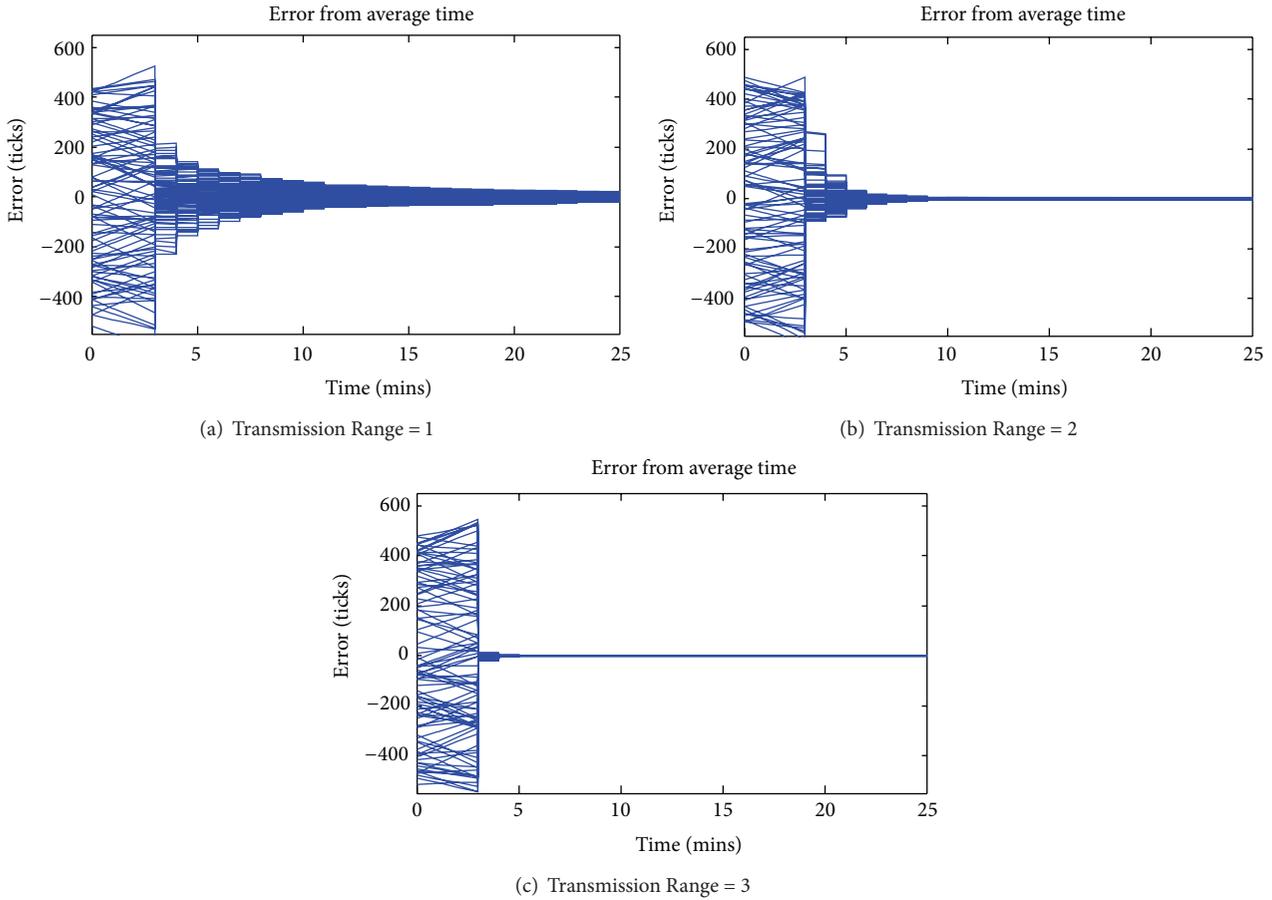


FIGURE 5: Clock errors of 100 simulated nodes running the TSMA protocol in a 10 * 10 network with synchronization period of 1 min in transmission ranges of 1 (a), 2 (b), and 3 (c).

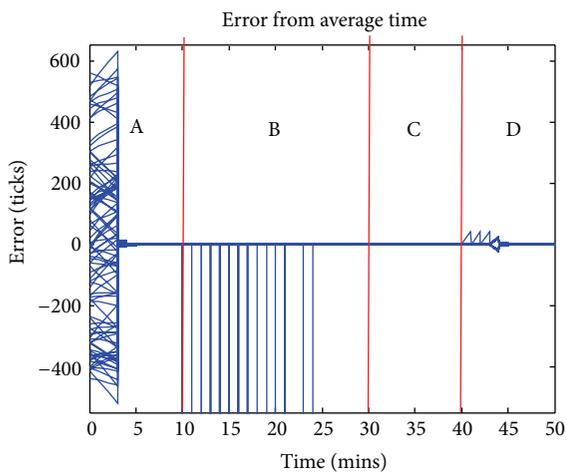


FIGURE 6: Clock errors of 100 simulated nodes running the TSMA protocol in a 10 * 10 network with synchronization period of 1 min and transmission range of 3. Region A: all nodes are on. Region B: 20% of the nodes are turned off and then turned on at random times. Region C: 20% of the nodes turned off their radio, and then turned on their radio after 5 synchronization periods. Region D: replace the node in the top left corner with one node having 40 PPM drift.

However, extensive work is still necessary to compare the performance of our proposed approach relative to FTSP [3] and other protocols over large scale multihop sensor network and over longer periods.

Acknowledgments

This paper is supported in part by Important National Science & Technology Specific Projects under Grants nos. (2010ZX 03006-002, 2010ZX03006-007), National Basic Research Program of China (973 Program) (no. 2011CB302803), and National Natural Science Foundation of China (NSFC) under Grant no. (61173132,61003307). The authors alone are responsible for the content of the paper.

References

- [1] H. Kopetz and W. Ochsenreiter, "Clock synchronization in distributed real-time systems," *IEEE Transactions on Computers*, vol. 36, no. 8, pp. 933-940, 1987.
- [2] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st*

- International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 138–149, Los Angeles, Calif, USA, November 2003.
- [3] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, “The flooding time synchronization protocol,” in *presented at Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, Baltimore, Md, USA, 2004.
 - [4] J. Elson, L. Girod, and D. Estrin, “Fine-grained network time synchronization using reference broadcasts,” *SIGOPS Operating Systems Review*, vol. 36, pp. 147–163, 2002.
 - [5] W. Su and I. F. Akyildiz, “Time-diffusion synchronization protocol for wireless sensor networks,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 384–397, 2005.
 - [6] L. Qun and D. Rus, “Global clock synchronization in sensor networks,” in *Proceedings of the Conference on Computer Communications -23th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM '04)*, pp. 564–574, March 2004.
 - [7] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, “Firefly-inspired sensor network synchronicity with realistic radio effects,” in *presented at the Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, San Diego, Calif, USA, 2005.
 - [8] L. Schenato and F. Fiorentin, “Average TimeSynch: a consensus-based protocol for clock synchronization in wireless sensor networks,” *Automatica*, vol. 47, pp. 1878–1886, 2011.
 - [9] M. K. Maggs, S. G. O’Keefe, and D. V. Thiel, “Consensus clock synchronization for wireless sensor networks,” *IEEE Sensors Journal*, vol. 12, pp. 2269–2277, 2012.
 - [10] D. L. Mills, “Internet time synchronization: the network time protocol,” *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, 1991.
 - [11] C. D. Godsil and G. Royle, *Algebraic Graph Theory*, Springer, 2001.
 - [12] R. Olfati Saber and R. M. Murray, “Consensus protocols for networks of dynamic agents,” in *Proceedings of the American Control Conference*, pp. 951–956, June 2003.



Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

