2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

# Time Synchronization in WSNs: A Maximum Value Based Consensus Approach

Jianping He, Peng Cheng, Ling Shi and Jiming Chen

*Abstract*— **This paper proposes a novel synchronization algorithm for wireless sensor networks (WSNs), the Maximum Time Synchronization (MTS), which is based on maximum value consensus approach. The main idea is to maximize the local information to achieve a global synchronization. Compared with the existing consensus-based synchronization protocols, the main advantages of our protocol include: i) a faster convergence speed such that the synchronization can be completed in a finite time; ii) simultaneous compensation for both the skew and the offset. We provide a rigorous proof of convergence to global synchronization and also give the upper bound of the convergence time. Moreover, the protocol is completely distributed, asynchronous, and robust against packet losses, nodes failure and the addition of new nodes. Some numerical examples are presented to demonstrate the efficiency of our protocol.**

## I. Introduction

Time synchronization is critical for many applications in wireless sensor networks, such as mobile target tracking, event detection, speed estimating, environment monitoring, etc. [1], [2]. It is essential for many applications in wireless sensor networks that all sensor nodes have a common time reference. Moveover, the clock synchronization also help to save energy in a WSN, as it provides the possibility to set nodes into the sleeping mode [3].

The authors of [4] propose a synchronization algorithm called Reference-Broadcast Synchronization (RBS) for one-hop time synchronization, where a node is selected as the reference node and then broadcasts a reference message to all the other nodes for synchronization. [5] aims to provide network-wide clock synchronization, and the authors propose a Timing-sync Protocol for Sensor Networks (TPSN). It first elects a root node and builds a spanning tree of the network, then the nodes are synchronized to their parents in the tree. However, the TPSN protocol can only compensate the relative clock offset but not the clock skew. Therefore, TPSN needs to send excessive messages for re-synchronization.

In order to overcome these shortcomings, [6] proposes the Flooding-clock synchronization Protocol (FTSP). The main idea is that the algorithm elects a root node and then the root node periodically floods its current time into the tree network. Using a Proportional-Integral control principle, [7] proposes a Feedback Based Synchronization (FBS) scheme to compensate the clock drift caused by both internal perturbation and external disturbance. It should be noted that all these three algorithms need a reference node or root node. Furthermore, TPSN , FTSP and FBS are tree-based synchronization, which are fragile to link or node failures. The detailed survey for clock synchronization protocols in wireless sensor networks can be found in [1] and [8].

On the other hand, consensus algorithms have been used for solving problems in autonomous networks including clock synchronization in WSNs. A consensus-based clock synchronization protocol has three main advantages. First, the consensus-based clock synchronization protocol can work in a distributed way. Therefore it does not require a tree topology or a root node as a reference [9], [10]; Second, based on the consensus algorithm, more accurate synchronized clocks, especially for neighboring nodes, may be obtained for the whole network [11]; Lastly, the consensus algorithms can compensate the skew differences among nodes.

Existing consensus distributed algorithms can be classified into two categories, synchronous algorithms, e.g., [12], [13], [14] and asynchronous ones, e.g., [15], [16], [17]. These two classes of consensus algorithms have been studied extensively for clock synchronization in wired or wireless ad-hoc and peer-to-peer networks, e.g., [3], [9]–[11], [18], [19]. Usually, the node adjusts its clock by a mutually agreed consensus value after each node has learned the clock values of all its neighbors. For instance, [11] proposes a Gradient Time Synchronization Protocol (GTSP) which is designed to provide synchronized clocks between neighbors, and this protocol is mainly based on a synchronous consensus algorithms [20]. Besides, [10] proposes an Average TimeSynch (ATS) protocol, which is built on an asynchronous consensus algorithm. The main idea is to average local information to achieving a global agreement on a specific quantity of interest.

Note that the converging speed of the time synchronization is a critical problem in practice, while most of existing consensus based protocols, which aim to reach an average value within the network, are time-consuming. Additionally, as pointed by [10], the exact value of the synchronized clock itself is not so important as long as the consensus has been achieved. Hence it is of great interest to develop a protocol which owns much faster converging time while maintaining the advantages of consensus.

The main contributions of this paper can be summarized as follows:

1. A novel time synchronization protocol is proposed,

Jianping He, Peng Cheng and Jiming Chen are with State Key Lab. of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China {jphe,pcheng,jmchen}@iipc.zju.edu.cn

Ling Shi is with the Department of Electric and Computer Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong eesling@ust.hk

978-1-61284-799-3/11/$26.00 ©2011 IEEE

which is based on a maximum value based consensus algorithm. The protocol is completely asynchronous and distributed, hence robust to packet losses, node failure, replacement or relocation.

2. We provide a rigorous proof of the convergence. Furthermore, we are able to prove that the algorithm will converge in a finite time and we present an upper bound analytically (of a linear complexity of the number of nodes), which is much lighter than those average based consensus algorithms.
3. Extensive simulations are conducted to prove the effectiveness for both static and dynamic graphs.

The rest of this paper is organized as follows. In Section II, the clock model for wireless sensor is introduced. The maximum valued based protocol for time synchronization is proposed In Section III, where a rigorous proof for its convergence is also provided as well as the analysis of its implementation of the MTS. The simulation results are shown in Section IV. Finally, Section V concludes this paper.

## II. Clock model

By refereing to [10], [11], Each sensor node $i$ in a WSN is equipped with a hardware clock, and its clock reading $\tau_i(t)$ at time $t$ is given by

$$\tau_i(t) = a_i t + b_i, \tag{1}$$

where $a_i$ is the hardware clock skew which determines the clock speed and $b_i$ is the hardware clock offset. It is reasonable to assume that each hardware clock skew $a_i$ satisfies

$$1 - p \le a_i \le 1 + p, \tag{2}$$

where $0 \le p < 1$ is a constant. It has been pointed out that $a_i$ and $b_i$ can not be computed as the absolute reference time $t$ is not available to the sensor nodes [10]. However, by comparing the local clock reading of one node $i$ with another local clock reading of node $j$, we can obtain a relative clock between them, given by:

$$\tau_i(t) = \tfrac{a_i}{a_j}\tau_j(t) + (b_i - \tfrac{a_i}{a_j}b_j) = a_{ji}\tau_j(t) + b_{ji} \tag{3}$$

The objective of clock synchronization in this paper is to synchronize all the nodes with respect to a common clock, given as:

$$\tau_v(t) = a_v t + b_v. \tag{4}$$

It is known that the value of hardware clock can not be adjusted manually, because other hardware components may depend on a continuously running hardware clock in WSNs [11]. Hence, logical clock is proposed to represent the synchronized time. Notice that the nodes can convert the hardware clock reading into a logical clock value, thus consider a logical clock value $L_i(t)$ which is a linear function of hardware clock $\tau_i(t)$,

$$L_i(t) = \hat{a}_i\tau_i(t) + \hat{b}_i = \hat{a}_i a_i t + \hat{a}_i b_i + \hat{b}_i \tag{5}$$

where $\hat{a}_i$ is the relative logical clock skew and $\hat{b}_i$ is the clock offset between the hardware clock and the logical clock. There always exist $(\hat{a}_i, \hat{b}_i)$ for each logical clock value $L_i(t)$ such that

$$L_i(t) = \tau_v(t), \ \ i = 1, 2, ..., N \tag{6}$$

where $N$ is the total number of nodes in WSN. Therefore, our goal in this paper is to design a consensus-based clock synchronization algorithm to find $(\hat{a}_i(k), \hat{b}_i(k))$ to each node $i$ in a WSN, such that

$$\begin{cases} \lim_{k\to\infty} \hat{a}_i(k)a_i = a_v, \\ \lim_{k\to\infty} \hat{a}_i(k)b_i + \hat{b}_i(k) = b_v, \end{cases} \tag{7}$$

where $a_v$ and $b_v$ are two constants. As pointed by [10], the real values of $(a_v; b_v)$ are not important; rather, it is important that all clocks converge to one common virtual reference clock, and the final parameters $(a_v; b_v)$ only depends on the initial condition and the communication topology of the WSN.

**Remark 1**

*Linear clock model is widely used for time synchronization in WSNs [9], [10], [19]. It should be noticed that $a_i$ and $b_i$ may be slowly time-varying in practice. However, as long as the synchronization algorithm can achieve certain accuracy in a short time, we can restart the algorithm once the synchronization error exceeds a given bound.*

## III. Main Results

We model a dynamic wireless sensor network (WSN) as a undirected graph $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$ at each time $t$, where $\mathcal{V} = \{i : i = 1, ..., N\}$ is the set of nodes (sensors) and $\mathcal{E}(t) \subseteq \{(i, j) : i, j \in \mathcal{V}, i \ne j\}$ represents the set of communication links (edges) at time $t$. The set of neighbors of node $i \in \mathcal{V}$ is denoted by $\mathcal{N}_i = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$. Throughout this paper, we assume the graph $\mathcal{G}(t)$ of WSN satisfies the following two assumptions:

**Assumption 1** There exists an integer $B \ge 1$ such that the undirected graph

$$(\mathcal{V}, \mathcal{E}(kB) \cup \mathcal{E}(kB + 1) \cup ... \cup \mathcal{E}((k + 1)B - 1))$$

is connected for all nonnegative integers $k$ [13].

**Assumption 2** For any edge, the time of this edge stays in graph is no less than a small constant $T_0$.

It is assumed that there is no transmission or reception delay, and the process and measurement noise is not taken into account for simplicity of statement.

### A. Synchronization Algorithm

From (7), it can be observed that it is necessary to synchronize both the clock skew and offset. Most of the existing algorithms consider these two parts separately [10], [11], which means that in their algorithms the offset compensation starts after the skew compensation has completed.

Two key parameters are introduced before we present the details of the algorithm. In WSNs, it is not difficult to set a constant $T$ for each node in advance that each node broadcasts its message to all its neighbors by a $T$ based on its own hardware clock.

**Definition 1**

*Give a constant period, $T$, the preudo-period $T_i$ is defined as [10],*

$$T_i = \frac{T}{a_i}, i \in \mathcal{V}. \tag{8}$$

Since the clock skew $a_i, i \in\in \mathcal{V}$ are slightly different in general, the preudo period $T_i$ would differ for each node. The second key parameter is called relative skew, which is defined as follows:

**Definition 2**

*Assume $(\tau_i(t_1), \tau_j(t_1))$ and $(\tau_i(t_2), \tau_j(t_2))$ are two pairs hardware clock reading of nodes $i$ and $j$. The relative skew $a_{ij}$ is defined as*

$$a_{ij} = \frac{\tau_j(t_2) - \tau_j(t_1)}{\tau_i(t_2) - \tau_i(t_1)}, i, j \in \mathcal{V}, \tag{9}$$

*where $t_1 \neq t_2$.*

Since each hardware clock model is a linear function of $t$, the relative skew $a_{ij}$ satisfies $a_{ij} = \frac{a_j}{a_i}$. Note that each hardware clock skew $a_i$ is time-invariant in model (1), so the relative skew $a_{ij}$ can be obtained directly by (9). In fact, when a node $i$ receive time message $\tau_j(t_1)$ from node $j$, it reads its current clock reading $\tau_i(t_1)$ and stores $(\tau_i(t_1), \tau_j(t_1))$; meanwhile when the node $i$ receives the time message from node $j$ at the second time, the relative skew $a_{ij}$ can be obtained from (9) based on these two pairs of historical records.

**Algorithm 1** : *Maximum Time Synchronization*

1: Given the initial conditions $\hat{a}_i = 1$ and $\hat{b}_i = 0$ for $i = 1, 2, ..., N$, set the common broadcast period $T$ to each node.
2: If the current hardware clock reading of node $i, i \in \mathcal{V}$ satisfies equation $\frac{\tau_i(t)}{T} \in \mathcal{N}^+$ (where $\mathcal{N}^+$ is the set of positive integer), then the node $i$ broadcasts its local hardware clock reading $\tau_i(t)$ and $(\hat{a}_i(t), \hat{b}_i(t))$ to its neighbors.
3: If a node $i \in \mathcal{V}$ receive the packet from its neighbor node $j$ at time $t_1$, then record its local hardware clock reading $\tau_i(t_1)$ and record the packet information as $[\tau_j(t_1), (\hat{a}_j(t_1), \hat{b}_j(t_1))]$.
4: If the node $i$ has a historical record $[\tau_i(t_0), \tau_j(t_0)]$, then compute $a_{ij}$ by

$$a_{ij} = \frac{\tau_j(t_1) - \tau_j(t_0)}{\tau_i(t_1) - \tau_i(t_0)} \tag{10}$$

and compute $d_i$ by $d_i = \frac{a_{ij}\hat{a}_j}{\hat{a}_i}$. After this delete the record $[\tau_i(t_0), \tau_j(t_0)]$.
5: If $d_i > 1$, then

$$\begin{aligned} \hat{a}_i &\leftarrow a_{ij}\hat{a}_j, \\ \hat{b}_i &\leftarrow \hat{a}_j(t_1)\tau_j(t_1) + \hat{b}_j - a_{ij}\hat{a}_j\tau_i(t_1); \end{aligned} \tag{11}$$

If $d_i = 1$, then

$$\hat{b}_i \leftarrow \max_{k=i,j}\{\hat{a}_k(t_1)\tau_k(t_1) + \hat{b}_k\} - \hat{a}_i\tau_i(t_1). \tag{12}$$

6: Store the time information $[\tau_i(t_1), \tau_j(t_1)]$.

In our algorithm, each node $i$ broadcasts its current hardware clock reading $\tau_i(t)$, the current skew compensation $\hat{a}_i$ and offset compensation $\hat{b}_i$ to its neighbor nodes by its own period. However, it is not assumed that the message is guaranteed to be received each time. When the nodes receive these messages from their neighboring nodes, they will adjust their logical clocks accordingly. Eventually, by the iteration of the algorithm, the logical clock of all nodes will equal to the maximum hardware clock, i.e., maximum-consensus. We therefore call the synchronization algorithm Maximum Time Synchronization (MTS). Through our MTS algorithm, the skew and offset compensation can be applied simultaneously and completed at the same time.

**Remark 2**

*In MTS, the common broadcast period $T$ should satisfy $\frac{T}{1-p} \leq T_0$ to ensure that each node has at least one broadcast when it contacts with the other nodes. Otherwise, some sensor nodes of the WSNs may not able to receive packets from the other sensor nodes all the time, which makes the global clock synchronization impossible.*

**Remark 3**

*For an arbitrary node $i$, its hardware clock reading should satisfy $\tau_i(t_k^i) = kT$, where $t_k^i$ denotes the reality time at $k$-th broadcast instant. Thus, we have*

$$t_k^i = \frac{kT - b_i}{a_i}, i \in \mathcal{V}, \tag{13}$$

*which in turn means $T_i = t_{k+1}^i - t_k^i = \frac{T}{a_i}$. Therefore, (8) is satisfied all the time.*

### *B. Convergence of MTS*

Before proving the convergence of our algorithm, we would like to first introduce some mathematical tools. Define $a_{max} = \max_{i\in\mathcal{V}} a_i$. We use $\mathcal{V}_{max}$ to denote the set of these nodes whose clock skews are equal to $a_{max}$. Define $b_{max} = \max_{i\in\mathcal{V}_{max}} b_i$. Let $a_v = a_{max}$ and $b_v = b_{max}$, then the common clock (4) then can be rewritten as:

$$\tau_v(t) = a_v t + b_v = a_{max}t + b_{max}. \tag{14}$$

Let $\mathcal{V}_v(t)$ be the set of nodes whose logical clock skew and offset are equal to $a_{max}$ and $b_{max}$ at time $t$, respectively. Clearly, the logical clock of nodes in $\mathcal{V}_v(t)$ are equivalent to the common clock $\tau_v(t)$ (14). Since the initial condition satisfies $\hat{a}_i = 1$ and $\hat{b}_i = 1$ for $i \in \mathcal{V}$ in MTS, according to the definition of $\mathcal{V}_v(t)$, it is clear to see that there will be at least one node whose hardware clock equal to the common clock at initial time, i.e., $\mathcal{V}_v(0) \neq \varnothing$.

Let $N_v(t)$ be the number of nodes belonging to the set $\mathcal{V}_v(t)$ at time $t$. Define a function $V(t)$ as follows

$$V(t) = N - N_v(t). \tag{15}$$

Since $\mathcal{V}_v(0) \neq \varnothing$, then we have $V(0) \leq N - 1$. The following result on $V(t)$ can be obtained.

**Lemma 1**

*$V(t) = 0$ if and only if $L_i(t) = \tau_v(t)$ for $\forall i \in \mathcal{V}$.*

**Proof:** From (15), $V(t) = 0$ if and only if $N_v(t) = N$, which is equivalent to that $i \in \mathcal{V}_v(t)$ if $i \in \mathcal{V}$. By the definition of $\mathcal{V}_v(t)$, it is equivalent to $L_i(t) = \tau_v(t), \forall i \in \mathcal{V}_v(t)$. The proof is completed. ■

Lemma 1 is equivalent to that $V(t) = 0$ if and only if $\mathcal{V}_v(t) = \mathcal{V}$.

**Theorem 1**

*Let the common synchronization period $T$ satisfy $\frac{T}{1-p} < T_0$, by using Algorithm MTS, the skew and offset will converge and satisfy*

$$\begin{cases} \lim_{k\to\infty} \hat{a}_i(k)a_i = a_v, \\ \lim_{k\to\infty} \hat{a}_i(k)b_i + \hat{b}_i(k) = b_v, \end{cases} \tag{16}$$

*where $a_v = a_{max}$ and $b_v = b_{max}$.*

**Proof:** By Lemma 1 and according to the definition of $\tau_v(t)$ in (14), it is obvious that once $V(t) = 0$ the skew and offset of all logical clocks would equal to $a_{max}$ and $b_{max}$ accordingly. Hence, it is enough to prove

$$\lim_{k\to\infty} V(k) = 0 \tag{17}$$

We first prove $V(t)$ is non-increasing. For any arbitrary node $i \in \mathcal{V}_v(t)$, its skew $\hat{a}_i(t)a_i$ and offset $\hat{a}_i(t)b_i + \hat{b}_i(t)$ equal to $a_{max}$ and $b_{max}$, respectively. Hence $\hat{a}_i(t)a_i \geq \hat{a}_j(t)a_j$ holds for $\forall j \in \mathcal{V}$, which means $\frac{a_{ij}\hat{a}_j}{\hat{a}_i} \leq 1$ hold for $\forall j \in \mathcal{V}$. In other words, this node $i$ will not update its logical clock skew any more as $d_i \leq 1$ will always hold for $k \geq t$. And, the equation $\hat{a}_i(t)b_i + \hat{b}_i(t) = b_{max}$ ensures that the node $i$ has the largest logical clock offset by the definition of $\mathcal{V}_v(t)$, which is equivalent to $L_i(t) = \tau_v(t), \forall i \in \mathcal{V}_v(t)$. Therefore, each node $i$ in $\mathcal{V}_v(t)$ will keep its logical clock skew and offset, which means that $N_v(t)$ is non-decreasing. Thus $V(t)$ is non-increasing.

Assume $V(t) > 0$ at time $t$ (where $t = mB, m \in \{0, \mathcal{N}^+\}$). Then, there is at lease one node in the network which is not in the set $\mathcal{V}_v(t)$. Since Assumption 1 guarantees the network is connected during any time interval $B$. Thus in time interval $[t, t + B]$, there will be at lease an edge (say $e(t)$) existing between the node sets $\mathcal{V}_v(t)$ and $\mathcal{V} - \mathcal{V}_v(t)$, and Assumption 2 can ensure $e(t)$ stays in the network more than $T_0$ ; and from condition $\frac{T}{a_i} \leq \frac{T}{1-p}, i \in \mathcal{V}$, we can get $T_i < T_0, i \in \mathcal{V}$. Then, there is at least one packet transmitted from one node in set $\mathcal{V}_v(t)$ to one node in $\mathcal{V} - \mathcal{V}_v(t)$ in time interval $[t, t + B]$ by communication link $e(t)$. Similarly, in the next time interval $[t+B, t+2B]$, there is also at least one packet can be transmitted from one node in set $\mathcal{V}_v(t+B)$ to one node in $\mathcal{V}-\mathcal{V}_v(t+B)$ by the edge $e(t + B)$ which connects $\mathcal{V}_v(t + B)$ and $\mathcal{V} - \mathcal{V}_v(t + B)$. Hence, in any a time interval $[t+kB, t+(k+1)B]$ for $k = 0, 1, ...$, there is at least one packet transmitted from one node $i$ in set $\mathcal{V}_v(t+kB)$ to one node $j$ in $\mathcal{V}-\mathcal{V}_v(t+kB)$ by an edge $e(t+kB)$. Since the number of nodes in both $\mathcal{V}_v(t)$ and $\mathcal{V} - \mathcal{V}_v(t)$ are finite, then there exists a $k_0$ such that node $i$ in $\mathcal{V} - \mathcal{V}_v(t)$ receives packets from same node $j$ in $\mathcal{V}_v(t)$ at time $t + k_0B$ for the second time. Then, according to MTS, the node $i$ will update its logical clock and then the node $i$ and $j$ will have the same logical clock, which means that node $i$ belongs to $\mathcal{V}_v(t + k_0B)$. Thus $N_v(t + k_0B) = N_v(t) + 1$, i.e., $V(t + k_0B)$ will be satisfied $V(t + k_0B) = V(t) - 1$.

Notice $V(0) \leq N - 1$ is finite, therefore $\lim_{k\to\infty} V(k) = 0$. ■

**Theorem 2**

*If the common synchronization period $T$ satisfies $\frac{2T}{1-p} < T_0$ and the skew and offset are updated using MTS, then the convergence time of the MTS satisfies:*

$$T_{con} \leq B(N - 1), \tag{18}$$

*where $T_{con}$ is the convergence time of the MTS algorithm.*

**Proof:** From the proof of the Theorem 1, we known $V(t)$ is a non-increasing function. Assume $V(t) > 0$ at time $t$ (where $t = mB, m \in \{0, \mathcal{N}^+\}$). Then, at each time interval $[t, t + B]$, Assumption 1 ensures that there is at least an edge $\{i, j\}$ which connects one node $i, i \in \mathcal{V} - \mathcal{V}_v(t)$ and node $j, j \in \mathcal{V}_v(t)$. Assumption 2 ensures taht this edge $\{i, j\}$ stays in the network for more than $T_0$. From $\frac{2T}{1-p} < T_0$, each real synchronization period $T_l, l \in \mathcal{V}$ satisfies $T_l < \frac{T_0}{2}$. Hence, the use of MTS, the node $i$ and $j$ can transmit messages to each other by using edge $\{i, j\}$ at least twice between the time interval $[t, t + B]$, which means that the node $i$ can set its logical clock equal to the logical clock of node $j$ before time $t + B$. Hence, we have $i \in \mathcal{V}_v(t + B)$, which means $V(t + B) = V(t) - 1$. Since $V(0) \leq N - 1$, $V((N - 1)B) = V(0) - N + 1 \leq 0$.

Therefore, $T_{con} \leq B(N - 1)$ from Lemma 1. ■

**Remark 4**

*In Theorem 1 and Theorem 2, Assumption 1 is used to ensure that there are nodes in the set $\mathcal{V} - \mathcal{V}_v(t)$ which can receive packets from the nodes in set $\mathcal{V}_v(t)$ at each time interval. This can be guaranteed by a strongly connected communication graph. Therefore, the condition Assumption 1 can be replaced by*

***Assumption 1'*** *There exists an integer $B \geq 1$ such that the undirected graph*

$$(\mathcal{V}, \mathcal{E}(kB) \cup \mathcal{E}(kB + 1) \cup ... \cup \mathcal{E}((k + 1)B - 1))$$

*is strongly connected for all nonnegative integers k [13].*

### C. The Analysis of Implementation of MTS

In this section, we will compare our approach with some typical consensus based time synchronization methods. It is not difficult to see that MTS algorithm is distributed, as every node updates its clock parameters based on the information received from its neighbor nodes only. In MTS algorithm, given a common synchronization period $T$ to all nodes in WSNs which is the same as ATS in [10], each node $i$ broadcasts packets based on $T_i$ accordingly, which can ensure that the algorithm is almost asynchronous. Furthermore, MTS is also robust to the addition of new nodes, nodes failure, and packet drops. However, these may affect the convergence time of the algorithm.

Note that different from the common consensus algorithms, the basic idea of our approach is to drive the logical clocks to the maximum value among all nodes so that the network can realize time synchronization as the final clock itself is not important. In the algorithm, each node $i$ periodically broadcasts a packet containing its local hardware clock reading $\tau_i(t)$ and its current relative logical clock skew $\hat{a}_i(t)$ and the offset $\hat{b}_i(t)$, and it doesn't need any feedback information from its neighbors nodes. And, the skew and offset compensation can be completed in a

finite time. Compared with the recently consensus-based time synchronization algorithms, i.e., GTSP [11] and ATS [10], MTS has two advantages as follows:

1. Both the algorithms GTSP and ATS converge to global synchronization asymptotically, however, the algorithm MTS can converge to global synchronization in a finite time. Furthermore, the convergence time of both GTSP and ATS depends on the synchronization error, however, the convergence time of MTS does not.
2. For GTSP and ATS, the offset compensation has to be started after the skew compensation algorithm has been completed. However, The offset compensation and the skew compensation of MTS can be conducted at the same time, and both can be completed simultaneously.

Due to these two advantages, MTS has a much faster convergence speed than these two consensus-based algorithms.

Energy cost is a major concern in WSNs. As the computational energy cost is comparably small due to the simple algorithm, we will focus on the communication energy cost, which can be measured by the broadcasting times. Let $N_i^c$ be the number of broadcast which each node $i$ needs to guarantee that the algorithm converges. In the above subsection, we have obtained the upper bound of convergence time of the algorithm. Based on Theorem 2, we can get an upper bound of $N_i^c$, which is given by

$$N_i^c \leq \frac{B(N-1)}{T_i}, i \in \mathcal{V} \tag{19}$$

where $T_i$ is the same as defined in (8). Assume that every broadcast of all the nodes is with the same amount of energy $E$ and let $E_c$ be the total energy cost for the synchronization algorithm to convergence, then

$$E_c \leq E \sum_{i\in\mathcal{V}} \frac{B(N-1)}{T_i} = \frac{EB(N-1)}{T} \sum_{i\in\mathcal{V}} a_i \tag{20}$$

From (20), it is not difficult to see that enlarging common period $T$ can decrease the upper bound of $E_c$, but $T$ should satisfy $\frac{2T}{1-p} \leq T_0$. Therefore, one choice of $T$ is $\frac{T_0(1-p)}{2}$.

## IV. SIMULATION RESULTS

For the simulations, we set $\hat{a}(0) = 1, \hat{b}(0) = 1$ and $T = 1$, and assume each skew $a_i$ of the hardware clock is randomly selected from the set $[0.8, 1.2]$, the offset $b_i$ of each node $i$ is randomly selected from the set $[0, 0.4]$. Additionally, we define two functions as follows:

$$\begin{aligned} d_s(t) &= \max_{i\in\mathcal{V}}(\hat{a}_i(t)a_i) - \min_{i\in\mathcal{V}}(\hat{a}_i(t)a_i) \\ d_o(t) &= \max_{i\in\mathcal{V}}(\hat{a}_i(t)b_i + \hat{b}_i(t)) - \min_{i\in\mathcal{V}}(\hat{a}_i(t)b_i + \hat{b}_i(t)) \end{aligned} \tag{21}$$

where $d_s(t)$ and $d_o(t)$ denote the maximum difference of logical skew and of logical offset between any two nodes, respectively. It is clear that the global time synchronization is reached if and only if $d_s(t) = 0$ and $d_o(t) = 0$.

In the following results, we will compare our synchronization algorithm MTS with ATS in [10], since ATS is a typical distributed and consensus-based synchronization algorithm.

Consider a static ring graph with $N = 30$ at first. It is well recognized that the distributed consensus-based algorithms often converges slowly for a ring graph. In Fig.1, it can

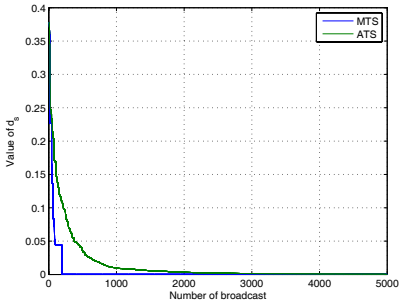


Fig. 1. Comparison between MTS and ATS in static network.

be observed that the skew synchronization is reached by 192 for MTS, while ATS takes more than 3200 to obtain a comparable accuracy. Additionally, from Fig. 2, it can be

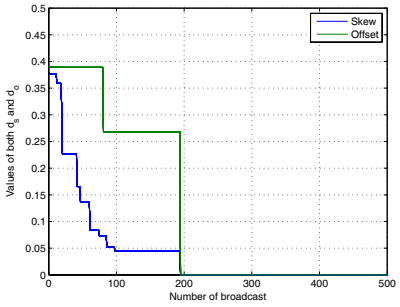


Fig. 2. Comparison between skew and offset using MTS in static network.

seen that for MTS, the skew and offset synchronization can be achieved at the same time, i.e., 100 times of tests, and the average iteration convergence time for MTS is 212, while ATS takes more than 4257 to ensure $d_s(t) \leq 10^{-4}$. Fig. 3

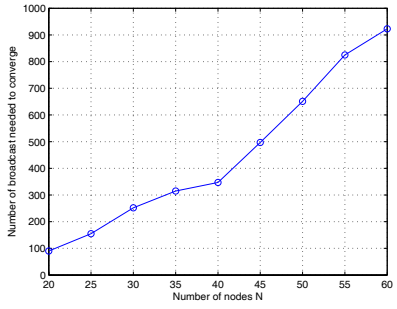


Fig. 3. The convergence time of MTS.

shows how MTS converges along with the the number of sensors in the network, which proves the scalability of our algorithm.

Consider a dynamic graph with $N = 50$. Assume that the nodes are randomly deployed in an $1 \times 1$ areas at initial time 0, and the maximum communication range of each node is $\sqrt{0.1}$. We assume that each node may randomly change its position once in every $20T$ until the convergence of one algorithm has been reached. Fig. 4 shows the initial graph and the final graph of the network. Similarity, as shown in Fig.5, $d_s(t) \leq 10^{-4}$ for $t \geq 431$ by ATS, however the $d_s(t)$

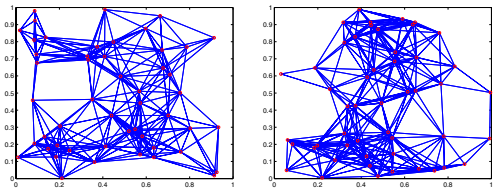

Fig. 4. The dynamical networks: the initial graph and final graph.

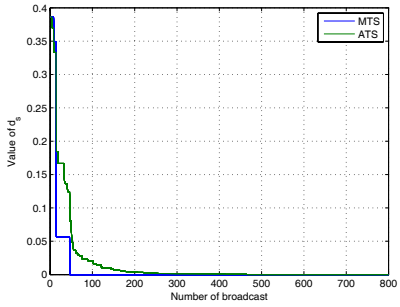


Fig. 5. Comparison between MTS and ATS in dynamical network.

of MTS decreases to zero when the iteration time $t \geq 46$. Thus, the convergence speed of MTS is much faster than that of ATS in the dynamical network. Furthermore, Fig. 6

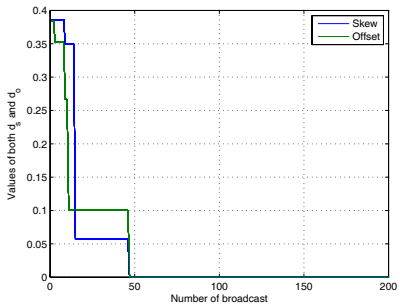


Fig. 6. Comparison between skew and offset using MTS in dynamic network.

shows that the $d_o(t)$ and $d_s(t)$ of MTS do converge to 0 at the same time $t = 46$ in the dynamical network, which shows the advantage of MTS also holds for dynamical network. The average time for MTS is around 47, and is about 545 for ATS to ensure $d_s(t) \leq 10^{-4}$.

## V. CONCLUSION

In this paper, we present a new time synchronization algorithm for WSN, the Maximum Time Synchronization (MTS) protocol, which conducts both the skew and offset compensations simultaneously. The main idea is to drive all the clocks to the maximum value among the network. It is proved that the MTS algorithm is guaranteed to converge within a finite time, which is much lighter than the average based consensus algorithms. The MTS algorithm is fully distributed, asynchronous and robust to dynamic network topologies. Extensive simulations demonstrate the effectiveness of our approach. Future directions include extending the idea to more complicated models and experimental validation of the results.

## ACKNOWLEDGEMENT

This work by J. He, P. Cheng and J. Chen is partially supported by the Natural Science Foundation of China (NSFC) under Grants 61004060 and 60974122, Joint Funds of NSFC-Guangdong under Grant U0735003, the Natural Science Foundation of Zhejiang Province under Grant R1100324. The work by L. Shi is partially supported by HKUST direct allocation grant DAG11EG06G.

## REFERENCES

[1] B. Sundararaman, U. Buy, and A. D. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3(3):281–323, 2005.

[2] S. He, J. Chen, D. Yau, H. Shao, and Y. Sun. Energy-efficient capture of stochastic events by global- and local-periodic network coverage. In *Proceedings of ACM MobiHoc*, pages 155–164, 2009.

[3] Q. Li and D. Rus. Global clock synchronization in sensor networks. In *Proceedings of IEEE INFOCOM*, pages 564–574, 2004.

[4] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of ACM OSDI*, 2002.

[5] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of SenSys 03*, 2003.

[6] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *Proceedings of SenSys 04*, 2004.

[7] J. Chen, Q. Yu, Y. Zhang, H.-H. Chen, and Y. Sun. Feedback based clock synchronization in wireless sensor networks: A control theoretic approach. *IEEE Transactions on Vehicular Technology*, 59(6):2963–2973, 2010.

[8] Y. R. Faizulkhakov. Time synchronization methods for wireless sensor networks: A survey. *Programming and Computing Software*, 33(4):214–226, 2007.

[9] C.D. Liao and P. Barooah. Time-synchronization in mobile sensor networks from difference measurements. In *Proceedings of CDC*, 2010.

[10] L. Schenato and F. Fiorentin. Average timesynch: a consensus-based protocol for time synchronization in wireless sensor networks. *Automatica*, 47(9):1878–1886, 2011.

[11] S. Philipp and W. Roger. Gradient clock synchronization in wireless sensor networks. In *Proceedings of IPSN*, 2009.

[12] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[13] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis. On distributed averaging algorithms and quantization effects. *IEEE Transactions on Automatic Control*, 54(11):2506–2517, 2009.

[14] J. Chen, X. Cao, P. Cheng, Y. Xiao, and Y. Sun. Distributed collaborative control for industrial automation with wireless sensor and actuator networks. *IEEE Transactions on Industrial Electronics*, 57(12):4219–4230, 2010.

[15] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

[16] P. Frasca, R. Carli, F. Fagnani, and S. Zampieri. Average consensus by gossip algorithms with quantized communication. In *Proceedings of IEEE CDC*, 2008.

[17] A. Kashyap, T. Basar, and R. Srikanta. Quantized consensus. *Automatica*, 43(7):1192–1203, 2007.

[18] A. Giridhar and P. R. Kumar. Distributed clock synchronization over wireless networks: Algorithms and analysis. In *Proceedings of IEEE CDC*, pages 4915–4920, 2006.

[19] N. Marechal, J.B. Pierrot, and J.M. Gorce. Fine synchronization for wireless sensor networks using gossip averaging algorithms. In *Proceedings of IEEE ICC*, pages 4963–4967, 2008.

[20] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.