

An Accurate On-demand References Broadcast Synchronization Protocol for Wireless Sensor Network

¹ Cheng Fang Zhen, ^{1,2} Wenyi Liu, ² Yulong Hou

¹ Key Laboratory of Instrumentation Science & Dynamic Measurement, North University of China, Ministry of Education, TaiYuan 030051, PR China

² National Key Laboratory for Electronic Measurement Technology, TaiYuan, Shanxi 030051, PR China

¹ Tel.: 13994215981, fax: 0351-3558768

¹ E-mail: zcf@nuc.edu.cn

Received: 28 April 2013 / Accepted: 19 July 2013 / Published: 31 July 2013

Abstract: Recent advances in technology have made low cost and low power wireless sensors a reality. Time synchronization may play a key role in wireless sensor networks. In this paper, The AORBS (accurate On-demand Reference Broadcast Synchronization Protocol) mainly made two improvements on the basis of typical RBS algorithm. Firstly, the node clock is not synchronized with the natural frequency at all time, and the synchronization algorithm will take effect only after an event of interest has happened. Waste is reduced since the system avoids performing synchronization at times when it is not necessary. Furthermore the algorithm achieves the drift compensation by using Taylor expansion but not the linear regression method, greatly reducing the computation cost and improving the synchronization accuracy. As shown in our experimental results: the improved algorithm is more efficient and can be applied to a variety of scopes and types of wireless sensor networks. Copyright © 2013 IFSA.

Keywords: Wireless sensor network, RBS (Reference broadcast synchronization), On-demand, Taylor expand.

1. Introduction

Wireless Sensor Networks (WSN) are composed of a set of planned or ad-hoc deployed sensors that are sensitive to their surrounding environments and capable of communicating with each other through their wireless channels. WSN can be applied in many application scenarios [1], such as smart environments, inventory management, precision agriculture, battlefield surveillance, animal herds tracking, and civil infrastructure monitoring. Recently, there has been a continuing increase in popularity and interest among researchers in Wireless sensor networks as the technical foundation of the

Smart Earth and Internet of Things era. Time synchronization is an important supporting technology of wireless sensor networks, which affects the operation and application of the sensor network protocol, such as tag data acquisition time, collaborative hibernation, positioning, and data fusion. These applications require accurate clock synchronization of the node in the network.

2. Related Works

Over the last decade, a number of time synchronization protocols have been proposed to

maintain the clocks in distributed systems [2]. NTP [3] has been the standard Internet time protocol because of its robustness, scalability, and flexibility. Synchronization protocol can be divided into three categories in accordance with transmitting-accepting model: the first is the sender-to-receiver unidirectional synchronization, in which the Flooding Time Synchronization Protocol (FTSP) [4] computes all delays in the packet transfer time to synchronize precisely both sender and receiver using only one broadcast and providing a global synchronization. The FTSP multi-hop synchronization algorithm includes both a leader selection (i.e., root node) and a flooding-like algorithm to propagate the timing information throughout the network. Multiple floorings can also be used to compute the clock drift; A similar approach is the Delay Measurement Time Synchronization (DMTS) [5], which differs from the FTSP on how the packet delay is estimated. Such algorithms are flexible, lightweight and energy-efficient, and they can achieve synchronization of all network nodes with high complexity and low accuracy.

The second category synchronization algorithm is based on the pair-wise bidirectional interaction, such as TPSN [6], Tiny-sync and Mini-sync [7], and the basic idea of TPSN is similar to NTP. It first organizes WSN into a hierarchical structure. Then synchronization algorithms based on two-way message exchange are performed along every edge of this tree-like structured network by taking the root node as the reference node. The implementation of TPSN is simple but not lightweight. Every node has to synchronize with its parent node by pair-wise message transmissions similar to NTP. Therefore, a lot of traffic over-heads are introduced into the networks.

The third is based on the receiver - receiver model such as RBS (References Broadcast Synchronization) [8]. RBS algorithm is a not-so-complicated classic time synchronization algorithm, which does not need to spend a large storage space, and is able to meet most of the time synchronization accuracy requirements. This algorithm is firstly proposed by J. Elson and K. Romer, at Hot Nets-I International Conference in August 2002 and has soon caused concerns in the field of sensor network research.

RBS [8] is derived from Post facto and it also keeps the time of neighboring nodes synchronized. One node periodically broadcasts to its neighbor reference beacons without explicit timestamps. The receivers use the beacon arrival time as a reference to compare their local clocks by exchanging the beacon receiving time. In this way, all the nodes know the clock offsets between each other. RBS requires a number of packet transmissions in order to exchange time information. It consumes a large amount of energy and computation power of sensor nodes. Moreover, RBS achieves drift compensation by using linear regression technique.

In this paper, we propose an accurate on-demand Reference Broadcast Synchronization algorithm,

abbreviated as AORBS. AORBS uses the same method of timestamp acquisition and estimation as the traditional RBS, taking the "busy running, idle sleeping" model to meet the on-demand reactive mechanism. This approach can greatly reduce the energy and calculating cost of wireless sensor network nodes to extend its life period. And moreover we use the Taylor expansion to achieve the drift compensation. Compared with the smallest linear regression method, it can reduce the spatial accumulative effect, save the data space, and improve the algorithm accuracy. All of these are shown in our simulation and experimental results.

3. Systems Model

3.1. Network Model

We assume a sensor network with N sensor nodes distributed in a random region. In general there is an internal crystal oscillator in any node [9, 10], which provides an almost stable vibration frequency to keep X's local continuously running. These nodes are connected into a distributed topology wireless sensor networks. We use a simple undirected graph $G=(V, E)$ to represent the distributed wireless sensor network. The set of vertices of the graph $V=\{1, 2, \dots, N\}$ represent the net. For any pair nodes (i, j) which can send and receive information to each other, we called it adjacent nodes. If $(i, j) \in E$, where E is the set of the margins, and (i, j) can represent a margin of the net. For any undirected graph, $(j, i) \in E$ is inevitable under the premise $(i, j) \in E$. An adjacent node of the $i \in V$ can be defined as follows:

$$S(i) = \{j \in V : (j, i) \in E\} t_x = a_x t + t_{0x} + \text{Drift}_x(t) \quad (1)$$

The metric of node i is represented as $d_i = |S(i)|$. We define the sequence vector of the graph G as $d = [d_1 d_2 \dots d_n]^T$. a_{ij} is the margin number from node i to node j. If the node i and node j are neighbors, then $a_{ij} = 1$, else $a_{ij} = 0$, we call $A = (a_{ij})_{N \times N}$ the adjacency matrix of graph A. We assume a static topology network and the undirected graph $G = (V, E)$ is a connected graph.

3.2. Clock Model

Clocks can be out of synchronization in two ways: drifting (clock skew-oscillator's frequency offset) or shifting (clock offset or phase offset) [9]. In the former case, they run at different rates. In the latter case, they run at the same rate, but their clock readings differ by a constant value—the shift

between the two clocks. Synchronizing clocks should solve both these two problems.

The clock rate of sensor node depends on the environmental temperature, the supply voltage, crystal quartz aging, and the conditions of the node's surrounding physical environment [11]. In fact, the relationships between node i 's clock and t_x the Coordinated Universal Time (UTC) t matches the following function:

$$t_x = a_x t + t_{0x} + \text{Drift}_x(t), \quad (2)$$

where a_x is the clock skew caused by the natural frequency differences between i 's oscillator and the ideal standard ones and t_{0x} is the initial time offset.

$\text{Drift}_x(t)$ is the rambled clock drift caused by the variations of environmental conditions. If the clock of i perfectly followed the UTC time t , a_x would be equal to 1 and both t_{0x} and Drift_x would be zero. Fig. 1 shows one case described by Eq.(2).

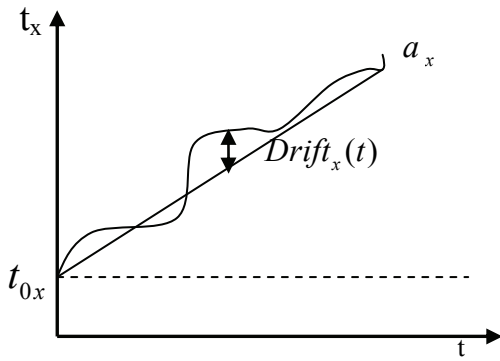


Fig. 1. Relationship between local clock and the UTC.

In the distributed wireless network, crystal oscillator is used to record the local time of each node i . The free-running output of the crystal oscillator can be represented in a cosine function as follows:

$$S_i(t) = \cos \Phi_i(t), \quad (3)$$

where

$$\Phi_i(t) = \Phi_i(0) + 2\pi / T_i + \zeta(t), \quad \Phi_i(t) \quad (4)$$

$\Phi_i(t)$ is the instantaneous phase, T_i is the period of the free oscillation, and $T_i = T_0 + \Delta T_i$, in which T_0 is the real period of oscillation, ΔT_i is random period deviation, $\Phi_i(0)$ represents initial phase, and $\zeta(t)$ is random phase noise.

$$t_i(n) = t_i(0) + nT_i + v(t) \quad (5)$$

The sequence $t_i(n)$ is defined as the time scale of the crystal oscillator, which is a discrete phase, and there's $\Phi_i(t_i(n)) = n \cdot 2\pi, n = 0, 1, 2, \dots$. Thus we can obtain the discrete uncoupled clock model:

$$t_i(n) = t_i(0) + nT_i + v(t) \quad (6)$$

where $t_i(0)$ is the initial phase, T_i is the period of the free oscillation, and $v(t)$ is the discrete random noise.

The clock synchronization is divided into the frequency synchronization and fully synchronization (i.e., the frequency and phase are synchronized). When the time parameter is large enough, if each time node has a common oscillating period T , which meets the following conditions:

$$\begin{aligned} T_i &= T_j = T, i \neq j, i, j = 1, 2, \dots, \\ t_i(n) &= t_j(n), i \neq j, j = 1, 2, \dots, N, n = 1, 2, 1/4, \end{aligned} \quad (7)$$

Then these network nodes are considered frequency synchronization. If each network node clock tick is equal, we can say the network is synchronization.

3.3. Model of Packet Delay

Another biggest enemy of precise time synchronization of sensor network is non-determinism [8]. We decomposes the packet delay into the following 6 components as shown in Fig. 2.

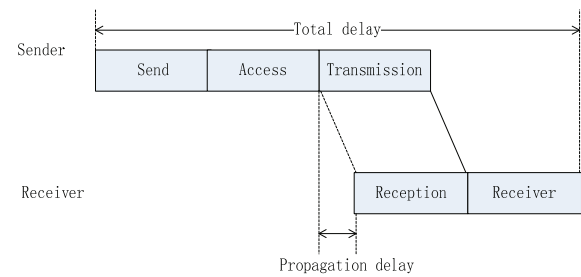


Fig. 2. Components of packet delay.

Send time: the delay which is spent in assembling a packet and delivering the packet to MAC layer in sender. It depends on the system call overhead of the operation system and the load of processor. It is nondeterministic.

Access time: the delay incurred by waiting for access to the wireless channel. It is the least deterministic part of packet delay.

Transmission time: the delay it takes for sender to transmit the packet bit by bit at the physical layer. It depends on the length of the packet and the transmission baud rate.

Propagation time: the delay it takes for one binary bit in packet to travel the wireless link from sender to receiver. It is deterministic and depends on the distance between the sender and the receiver.

Reception time: the delay it takes for the receiver to receive the packet. Similar to transmission time, it depends on the length of packet and the transmission baud rate. It may partly overlay with transmission time.

Receive time: the delay of processing the incoming packet and delivering it to the application layer in receiver. Its character is similar to that of send time.

4. Accurate On-demand References Broadcast Algorithm

For convenience, we first introduce some terms and assumptions used throughout this paper. A sequence of nodes $A_1, \dots, A_n$, is called a path from A_1 to A_n if A_{i-1} is in the communication range of A_i for each $1 < i \leq n$. A network is called connected if there is a path connecting every pair of nodes. For a given inter-node, we call the difference between the time of the event about to be sent by the inter-node and the time of the event about to be received by the inter-node as the residence time of the event in the inter-node. $T_{i,k}$ denotes the time of node i for receiving the k -th message.

4.1. AORBS Principle

As it is consumed in the last section, the network is clustered such that all the nodes in the same cluster can communicate with each other directly and with cluster heads. The execution of the algorithm is shown in Fig. 3.

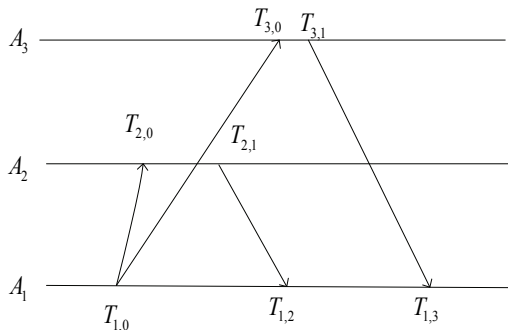


Fig. 3. The Fundamental Principle of AORBS.

AORBS's execution sequences are listed as follows:

- 1) Node A_1 gets the event timestamp from local time and broadcasts it to all nodes in the cluster;
- 2) A_i marks the arrival time of beacon packet in its own MAC layer, and calculates its residence time, and these two sums up to the node A_i 's local time;
- 3) Calculate the local clock time of A_i and A_{i-1} , and then send the clock offset back to node A_1 ;
- 4) The sink node A_1 counts the average of all the clock offset;
- 5) The sink node A_1 collaborates all of the drift by Taylor Expansion to obtain the Offset coefficient;
- 6) When events reach the sink node, the node executes the same sequence of step 2;
- 7) The sink node estimates the global time by adjusting the drift coefficient and statistics clock offset to synchronization.

We record two timestamps at both the sender and receiver sides when the message is being transmitted, as the method used in [12, 13]. Therefore, during the synchronization process using AORBS, a participating node will collect either four or two timestamps. Each inter-node I will collect four timestamps $T_{i,k}$ ($k=1,2,3,4$) successively. The first two timestamps $T_{i,k}$ ($k=1,2$) are collected during the phase when I receives the synchronization packet from its child node (prior node), while the last two timestamps $T_{i,k}$ ($k=3,4$) are collected in the phase when I sends the synchronization packet to its parent node (successor node). Since the event source A_0 does not experience the phase of receiving the synchronization packet, it only collects two timestamps $T_{i,k}$ ($k=3,4$). Similarly, as the sink node A_n does not experience the phase of sending the synchronization packet, it only collects two timestamps $T_{i,k}$ ($k=1,2$).

Let A_n be the sink node, and A_0 be the event source. Since the network forms a breadth-first search tree whose root is the sink node, there exists a unique sequence of nodes $A=A_0, A_1, \dots, A_i=A_0$, such that for every $0 < i \leq n$, A_i is the parent node of A_{i-1} . For simplicity, we use $T_{i,k}$ ($k=1,2,3,4$) to denote $T_{A_i,k}$ ($k=1,2,3,4$) for every $0 < i < n$; Moreover, let $T_{i,k} = C_i(T_{i,k})$ for every $0 < i < n$ and $1 \leq k \leq 4$; let $T_{0,k} = C_0(T_{0,k})$ for $k=3$ and 4 ; and let $T_{n,k} = C_n(T_{n,k})$ for $k=1$ and 2 . Next we describe the procedure in which A_0 is synchronized to A_n by using AORBS.

As soon as A_0 detects an event of interest, it first uses its own clock to timestamp the event, and then prepares to transfer a synchronization packet M_0 to other members in the cluster as described above. Before A_0 is about to send M_0 , it obtains its own clock value $T_{0,3}$ through the MAC layer, and uses $T_{0,4} - T_{0,3}$ to fill in M_0 's eta, where $T_{0,0} = C_0(t_0)$ is

the timestamp of the event. Before A_1 is about to receive M_0 , it obtains its own clock value $T_{1,1}$ through the MAC layer. When A_0 finishes sending the last bit of M_0 ·eta, it obtains its own clock value $T_{0,4}$ through the MAC layer, and uses $T_{0,4} - T_{0,3}$ to fill in M_0 ·eta.

Once A_1 finishes receiving the last bit of M_0 ·eta, it obtains its own clock value $T_{1,2}$ through the MAC layer. For every $0 < i < n$, the part of the synchronization procedure related to A_i is as shown in Fig. 3. Although A_i at the MAC layer obtains timestamps $T_{1,i}$ and $T_{2,i}$, it can only obtain the entire synchronization packet M_{i-1} at the application layer. Once A_i receives the entire synchronization packet M_{i-1} from A_{i-1} , it obtains a synchronization triple $(T_{0,4} - T_{0,3}, T_{1,i}, T_{2,i})$ as $M_{i-1} \cdot tt = T_{0,4} - T_{0,3}$. Let $\delta = T_{1,i} - T_{0,3}$. Then the event arrives at A_i from A_0 after δ_i delay. The problem of synchronization between A_i and A_0 can be reduced to finding the time transformation function $f_{i \rightarrow 0}$ such that $C_0(t - \delta_i) = f_{i \rightarrow 0}(C_i(t)), \forall t \geq t_{1,i}$. In theory $f_{i \rightarrow 0}$ can be constructed from the following parametric equation system:

$$\begin{cases} x = C_i(t) \\ y = C_0(t - \delta_i) \end{cases}, t_i^1 \leq t < +\infty \quad (8)$$

Since $x = C_i(t)$ is a strictly increasing function of t , $t = C_i^{-1}(x)$, therefore, $y = C_0(C_i^{-1}(x) - \delta_i)$, that is, $f_{i \rightarrow 0}(x) = C_0(C_i^{-1}(x) - \delta_i)$. By the parametric equation system (8) and the derivative formula for the function defined by it,

$$f'_{i \rightarrow 0} \Big|_{x=C_i(t)} = \frac{C'_0(t - \delta_i)}{C'_i(t)}, t_i^1 \leq t < +\infty \quad (9)$$

It is very reasonable from Eq.(9) and Eq.(11) to use the above $\tilde{f}_{i \rightarrow 0}$ to approximate $f_{i \rightarrow 0}$. Before A_n is about to receive the last synchronization packet, where $M_i \cdot tt = M_{i-1} \cdot tt$, it obtains its own clock value $T_{n,3}$ through the MAC layer, and uses $M_{i-1} \cdot eta + \Delta_i$ to fill in $M_i \cdot eta$ where $\Delta_i = \tilde{f}_{i \rightarrow 0}(T_{n,3}) - \tilde{f}_{i \rightarrow 0}(T_{n,1})$, that is,

$$\Delta_i = \frac{T_{0,4} - T_{0,3}}{T_{i,2} - T_{i,0}} \cdot (T_{i,3} - T_{i,1}) \quad (10)$$

Since Δ_i is approximately the time (In the local time of A_0) which elapsed from the event occurrence at physical time t_0 to the event arrival at A_n at physical time $t_{n,1}$, $C_0(t_{n,1}) - \Delta \approx C_0(t_0)$. Hence we can say that every node in the synchronization procedure using AORBS has finished by using 3 arithmetic operations only.

4.2. Drift Compensation Using Taylor Expansion

To avoid frequent resynchronization, many synchronization algorithms adopt drift compensation techniques, i.e. they estimate the ratio between the clock speed of the node to be synchronized and that of the reference node and then use this estimated value to compensate the synchronizing clock reading [14]. AORBS uses Taylor expansion to implement drift compensation. Next we show that drift compensation achieved by using Taylor expansion has better accuracy than drift compensation achieved by using linear regression, which is the most common technique used by current time synchronization algorithms.

Let $f(t)$ be the clock function of a node. The best uniform linear approximation of f on an interval $[a, b]$ is defined as a linear function $\bar{p}_1(x)$ such that for any linear function $p(x)$, $\max_{a \leq x \leq b} |\bar{p}_1(x) - f(x)| \leq \max_{a \leq x \leq b} |p(x) - f(x)|$. Chebyshev proved that the best uniform linear approximation exists and is unique [16]. From this definition, it is clear that the best uniform linear approximation is better than other linear approximations. Using Theorem 4.8 (the oscillating theorem) in [14], we can prove that the best uniform linear approximation of f on the interval $[a, b]$ is

$$\bar{p}_1(x) = \frac{f(a) + f(\xi)}{2} + \frac{f(b) - f(a)}{b - a} \cdot \left(x - \frac{a + \xi}{2}\right) \quad (11)$$

where $a < \xi < b$, such that $f'(\xi) = \frac{f(b) - f(a)}{b - a}$. Calculating the drift

compensation on the interval $[a, b]$ by using Taylor expansion is equal to using the line segment

$$\hat{p}_1(x) = f(a) + \frac{f(b) - f(a)}{b - a} \cdot (x - a) \quad \text{as}$$

substitute for the clock curve and using the slope of this line segment as the estimated value of the clock frequency. It can be easily seen that the line segment $\hat{p}_1$ obtained by the Taylor expansion drift compensation on the interval $[a, b]$ and the best

uniform linear approximation $\bar{p}_1$ on the interval $[a, b]$ are in parallel, and the distance between them is very small. Suppose that at physical times $a = x_1 < x_2 < \dots < x_{n-1} < b = x_n$ the values of the clock f are $y_1 \dots y_n$ respectively. Doing the drift compensation on the interval $[a, b]$ by using linear regression is equal to using the following line segment

$$p_1(x) = \frac{1}{n} \sum_{i=1}^n y_i + \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2} \cdot (x - \frac{1}{n} \sum_{i=1}^n x_i) \quad (12)$$

as substitute for the clock curve and using the slope of this line segment as the estimated value of the clock frequency. The probability of

$$\frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2} = \frac{f(b) - f(a)}{b - a}$$

is very small because $x_i (2 \leq i \leq n-1)$ has randomness. Since Taylor expansion drift compensation uses the slope of the best uniform linear approximation as the estimated value for the clock frequency, while linear regression drift compensation generally does not, the effect of the drift compensation achieved by using Taylor expansion is better than that of the drift compensation achieved by using linear regression.

4.3. Computational Complexity and Spatial Accumulative Effect

Time synchronization is typically based on one or several packet exchanges containing the timestamp [15]. For a given event, the communication cost of an on-demand time synchronization algorithm to transform the timestamp of the event from the local time of the source of the event to the local time of the sink node (we call it the communication cost of the algorithm with regard to the event, for short), is defined as the total number of the packet exchanges during the transformation process.

Let E be an event, and A be the source of E. Let n be the hop distance from event source A to the sink node. Therefore there exists a path with length n $A_0, A_1, \dots, A_n$, such that $A_0 = A$, and A_n is the sink node. In order to convert the timestamp of event E from the local time of event source A to the local time of the sink node by an on-demand

synchronization algorithm, in fact we must first use the algorithm to complete the time synchronization between event source A and the sink node. To achieve the synchronization between them, there must exist at least one synchronization packet that starts from event source A, after n packet exchanges and reaches the sink node or vice versa. Hence, the communication cost of the algorithm regarding event E is at least n. Therefore, it is straight forward to see that, for a given event, the communication cost of any on-demand time synchronization algorithm regarding the event is greater than or equal to the hop distance between the source of the event and the sink node. We will see from the description about AORBS that the communication cost of AORBS regarding any event is equal to the hop distance between the source of the event and the sink node. Communication is a primary source of energy consumption in WSN. Even though the communication cost of an on-demand synchronization algorithm regarding an event is a function of the hop distance between the source of the event and the sink node, sometimes we only mention the communication costs of algorithms regardless of any events. When in our experiments we compare the communication cost of two different on-demand synchronization algorithms, we always base the comparisons on the same event, although we do not mention this explicitly.

It is not difficult to see that the computational complexity of the first-order Taylor expansion whose monomial coefficient is replaced with the ratio of the increment of the function to the increment of the argument is 6 arithmetic operations. By the formula in [11], performing a linear regression requires $6N+9$ arithmetic operations, where $N (\geq 2)$ is the number of data points used in the linear regression. Hence, the computational complexity of any on-demand synchronization algorithm which achieves drift compensation by using linear regression technique is $N+1$ times greater than that of any algorithm which achieves drift compensation by using the first-order Taylor expansion technique. Since all existing on-demand synchronization algorithms which achieve drift compensation use linear regression and our algorithm AORBS achieves drift compensation by using the first-order Taylor expansion, the computational complexity of any existing on-demand synchronization algorithm which achieves drift compensation is $N+1$ times greater than that of AORBS. Because the computational complexity of AORBS is only 6 arithmetic operations, it has very low computational complexity, which is an important requirement considering the constrained processing power of WSN nodes.

For on-demand synchronization algorithms with a strong spatial accumulative effect using linear regression to achieve drift compensation, if the absolute value of the error in step one is ϵ , then the error in step two is approximately ϵ^2 , and in step three the error is approximately ϵ^3 . For on-demand synchronization algorithms without a strong spatial accumulative effect using Taylor expansion to

achieve drift compensation, if the absolute value of the error in step one is η , then the error in step two is approximately 2η , and in step three the error is approximately 3η . Hence, the error in step n is $O(n\eta)$. Although in the case of on-demand synchronization η is much smaller than ε , in large-scale networks the accuracy of algorithms that use Taylor expansion to achieve drift compensation and have a strong spatial accumulative effect is much lower than that of algorithms that use linear regression to achieve drift compensation.

5. Error Analysis

The estimated time of the RBS algorithm can be divided into two parts, receiving estimated time of the node and system clock error estimates. The node receiving estimated time ΔT_i is the average differences of numbers of receiving message time $T_i, T_j (i \in n, j \in n)$. Two nodes receiving differences message time error $d_i \forall i \in n$ can be calculated by:

$$\forall i \in n, j \in n; \Delta T_i = \text{offset}[i, j] = \frac{1}{m} \sum_{k=1}^m (T_{i,j} - T_{i,k}), \quad (13)$$

where n represents the n -th node, and m represents the m -th message. The system clock error for the node can be found by Taylor Expansion as follows:

$$\forall i \in n, j \in n; d_i = (T_{j,k} - T_{i,k}) - (a + bT_{i,k}) \quad (14)$$

Through the above analysis, the receiving time of the k -th message of each node $T_{i,k}$ play the decisive role. Let's assume $T_{1,k}$ as the time of node 1 for correctly receiving the k -th message, $T_{1,k}'$ as the time of node 1 for receiving the k -th message after the error happened, the difference of $T_{1,k}'$ and $T_{1,k}$ is constant $d_{1,k}$. Similarly, the receiving time of the k -th message of node 2 is $T_{2,k}$, $T_{2,k}'$ is the time of node 1 for receiving the k -th message after the error happened, and the difference of $T_{2,k}$ and $T_{2,k}'$ is a constant $d_{2,k}$. By formula (2), we can work out the error of receiving time difference ε between the two cases of errors and error-free, which can be shown as follows:

$$\varepsilon = \Delta T_{i,k}' - T_{i,k} = \frac{1}{m} \sum_{k=1}^m (d_{2,k} - d_{1,k}) \quad (15)$$

6. Performed Experiments

To evaluate the performance of the proposed algorithms we implemented and tested them on two sets of data points. Since a wireless sensor network was not available, we used an 802.11b multi-hop ad-hoc network. It is expected that wireless sensor networks will exhibit similar delay patterns as the ones encountered in this ad-hoc network. To mimic the data traffic in the wireless sensor network, we used background traffic in the ad-hoc network as well. For the first set of data points the two computers to be synchronized were neighbors (i.e. one hop away). For the second data set, the computers were five hops away. For both experiments a message probe was sent once a second for about 83 minutes thus resulting in almost 5000 data-point samples for each experiment. Each experiment was repeated 32 times to increase the confidence in the results. The statistics of the collected data-points are presented in Table 1. We will analyze the synchronization error of AORBS compared to three comprehensive time synchronization algorithms, namely TPSN [6], FTSP [4] and traditional RBS [8].

Table 1. Statistics of the data-points collected for the one hop and the five hops experiments.

	Min (ms)	Max (ms)	Ave. (ms)	Std. Dev. (ms)
One hop	3.0401	24.217	3.366	1.112
Five hops	16.073	177.744	31.540	12.874

In our first experiment, we observe 32 synchronization processes in which D always takes value 1 and N takes values 1–8. We use $X_{i,j} (i = 1, 2, 3, 4; j = 1, \dots, 8)$ to represent the average absolute errors of these synchronization processes, where i corresponds to one of the evaluated algorithms, and j corresponds to one of the 8 values of N . Since the clock frequency of every node is random, each $X_{i,j}$ is a random variable. We use $E(X_{i,j})$ to denote the mean of $X_{i,j}$, and use $I(X_{i,j})$ to denote the 95 % confidence intervals. We repeat this experiment 30 times. We use $X_{i,j}^{(k)} (k = 1, \dots, 30)$ to represent the k -th sample value of $X_{i,j}$. According to probability theory, we know that

$$E(x_{i,j}) = \frac{1}{30} \sum_{k=1}^{30} X_{i,j}^{(k)} \quad (16)$$

and

$$I(x_{i,j}) = [E(x_{i,j}) - y_{i,j}, E(x_{i,j}) + y_{i,j}] \quad (17)$$

where

$$y_{i,j} = \frac{2.045}{\sqrt{30}} \times \sqrt{\frac{1}{29} \times (\sum_{k=1}^{30} (X_{i,j}^{(k)})^2 - 30 \times E(x_{i,j})^2)} \quad (18)$$

The average absolute errors considering a 95 % confidence interval for each algorithm are drawn in Table 2. The elements of the AORBS row are $I(X_{1,j})$, $j = 1, \dots, 8$, and the other rows have a similar meaning. We draw the average absolute errors for each algorithm in Fig.4.

The points on the 'AORBS' curve are $(1, E(X_1, 1)) \sim (8, E(X_1, 8))$, where $(i, E(X_1, i))$ is a point with horizontal coordinate i and vertical coordinate $E(X_1, i)$ for every $1 \leq i \leq 8$, and the other curves have similar meaning.

When $N=1$, both TPSN and FTSP become linear. Since the accuracy of TPSN is low, the accuracy of both TPSN and FTSP is lower than that of RBS when

$N=1$. Because both AORBS and TPSN do not require performing linear regression, their accuracy does not change obviously as N grows. When $N=1$, FTSP uses TPSN to synchronize the event source to the sink node after they exchange two synchronization packets, while FTSP uses TPSN to synchronize the event source to the sink node immediately after the event occurred.

We only list the experiments analysis in multi-hops in the paper. From Fig. 4 we can say that the accuracy of TPSN is more than 10 times higher than that of FTSP. The biggest shortcoming of FTSP is its unsuitability in a multi-hop scenario. Since the elapsed time for synchronization is usually short, Taylor expansion has obvious advantage in comparison to linear regression in terms of accuracy. Because AORBS achieves drift compensation by Taylor expansion, while others use linear regression, thus the accuracy of AORBS is higher than that of them. From Fig. 4 we can say that the accuracy of AORBS is about 10 times higher than that of traditional RBS.



Fig. 4. Error analysis of four algorithms in multi-hops.

Table 2. Average absolute errors of experiment.

Algorithm	D							
	1	2	3	4	5	6	7	8
AORBS	[0.22, 0.41]	[0.52, 0.81]	[0.90, 0.94]	[0.69, 0.99]	[0.95, 0.99]	[0.46, 0.86]	[0.66, 0.89]	[0.34, 0.75]
TPSN	[1.67, 2.93]	[1.71, 2.88]	[1.76, 3.1]	[1.53, 2.75]	[1.75, 2.97]	[1.81, 3.09]	[1.67, 3.22]	[1.7, 3.01]
FTSP	[2.72, 4.78]	[1.05, 1.76]	[1.14, 2.01]	[1.0, 1.8]	[1.12, 1.93]	[1.16, 2.01]	[1.08, 2.1]	[1.1, 1.96]
TRBS	[5.88, 12.02]	[0.28, 0.63]	[0.15, 0.31]	[0.09, 0.18]	[0.1, 0.16]	[0.13, 0.24]	[0.09, 0.16]	[0.14, 0.22]

7. Conclusion

A light-weight synchronization algorithm is presented. The proposed algorithm is able to produce tight, and deterministic synchronization with only few message exchanges. While the algorithm is suitable for any type of network, it is especially useful in wireless sensor networks which are typically extremely constrained on the available computational power and bandwidth and have some

of the most exotic needs for high precision synchronization. The performance of the presented algorithms is verified with an experimental test bed. The experimental results match closely the theoretical expectations. The algorithm is presented and its performance is compared using real delay traces collected from a wireless ad-hoc network with other algorithms. The experimental results show that the most effective of the four algorithms produces results very close to the optimum (within 0.1 %) and thus is more preferable.

References

- [1]. Bharath, Sundararaman, Clock synchronization for wireless sensor networks: survey, *Ad-hoc Networks*, Vol. 3, No. 3, May 2005, pp. 281-323.
- [2]. Faranak, Heidarian, Analysis of a clock synchronization protocol for wireless sensor networks, *Theoretical Computer Science*, 413, pp. 2012, pp. 87-105.
- [3]. D. L. Mills, Network Time Protocol (Version 3) specification, implementation and analysis, Request for Comments (RFC) 1305 [Z].
- [4]. M. Maroti, B. Kusy, The flooding time synchronization protocol, Baltimore, MD, USA: in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, 2004, pp. 39-49.
- [5]. S. Ping, Delay measurement time synchronization for wireless sensor networks, Tech. Rep. IRB-TR-03-013, *Intel Research*, June 2003.
- [6]. S. Ganeriwal, R. Kumar, M. B. Srivastava, Timing-sync protocol for sensor networks, *Proceedings of the SENSYS'03*, Los Angeles, November 2003.
- [7]. M. L. Sichitiu, C. Veerarittiphan, Simple accurate time synchronization for wireless sensor networks, in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'03)*, New Orleans, LA, March 2003.
- [8]. Jeremy, Elson, Lewis, Girod, Fine-Grained Network Time Synchronization using Reference Broadcasts, *SIGOPS Operating Systems Review*, 36, SI, 2002, pp. 147-163.
- [9]. F. Y. Ren, S. Y. Dong, T. He, C. Lin, A time synchronization mechanism and algorithm based on phase lock loop, *Journal of Software*, 18, 2, 2007, pp. 372-380.
- [10]. Ill-KeunRhee, Jaehan Lee, Jang Subkim, Clock synchronization in wireless sensor networks: an overview, *Journal of Sensors*, 9, 1, 2009, pp. 56-85.
- [11]. B. van Voorst, Time synchronization in wireless sensor networks. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.6280&rep=rep1&type=pdf>
- [12]. B. Kusy, P. Dutta, P. Levis, M. Maroti, A. Ledeczi, D. Culer, Elapsed time on arrival: a simple and versatile primitive for canonical time synchronization services, *International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC)*, 2, 1, 2006.
- [13]. M. Maroti, B. Kusy, G. Simon, A. Ledeczi, The flooding time synchronization protocol, in *Proceedings of the 2nd Int'l Conf. on Embedded Networked Sensor Systems*, 2004, pp. 39-49.
- [14]. L. Schenato, G. Gamba, A distributed consensus protocol for clock synchronization in wireless sensor network, in *Proceedings of the 46th IEEE Conference on Decision and Control*, December 2007.
- [15]. J. Yick, B. Mukherjee, D. Ghosal, Wireless sensor network survey, *Computer Networks*, 52, 2008, pp. 2292-2330.
- [16]. B. Liu, F. Ren, J. Shen, H. Chen, Advanced self-correcting time synchronization in wireless sensor networks, *IEEE Communications Letters*, 14, 2010, pp. 309-311.

2013 Copyright ©, International Frequency Sensor Association (IFSA). All rights reserved.
(<http://www.sensorsportal.com>)

Universal Sensors and Transducers Interface (USTI)

for any sensors and transducers with frequency, period, duty-cycle, time interval,
PWM, phase-shift, pulse number output



- * Input frequency range:
0.05 Hz ... 9 MHz (144 MHz)
- * Selectable and constant relative error:
1 ... 0.0005 % for all frequency range
- * Scalable resolution
- * Non-redundant conversion time
- * RS232, SPI, I2C interfaces
- * Rotational speed, rpm
- * Cx, 50 pF to 100 µF
- * Rx, 10 Ω to 10 MΩ
- * Pt100, Pt1000, Pt5000, Cu, Ni
- * Resistive Bridges
- * PDIP, TQFP, MLF packages

Just make it easy !

<http://www.techassist2010.com/>
 info@techassist2010.com