Telecommun Syst
DOI 10.1007/s11235-008-9081-5

Ratio-based time synchronization protocol in wireless sensor networks

Jang-Ping Sheu · Wei-Kai Hu · Jen-Chiao Lin

Abstract Time synchronization plays a key role in the wireless sensor networks (WSNs). Time synchronization is realized by those messages that are time-stamped. But there are several delay times during transmission after time stamping. Most of them are uncertain and contribute directly to synchronization error. The uncertainties include send time, channel access time, transmission time, and receive time. In addition to the uncertainties, clock drift is also a main source of time synchronization error. In this paper, we present a time synchronization protocol that can be applied in the multi-hop WSNs. The proposed protocol estimates the clock drift between two nodes to keep them synchronized for duration after once synchronizing. It uses lower communication overhead and establishes more robust synchronization situations for all nodes in the network. By periodical re-synchronization, the un-synchronization conditions such as nodes failures or topology change can be easily overcome. We implement our protocol in the Berkeley MICAz platform. The experimenting scenarios are 5-node and 18-node multi-hop topologies, and the re-synchronization periods are 30-second and 300-second. The experiment results show that the average synchronization errors of all nodes run with our protocol are ranged within several micro-seconds which are better than the previous protocol.

Keywords Clock drift · Time synchronization · Wireless sensor networks

J.-P. Sheu (✉)
Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan
e-mail: sheujp@csie.ncu.edu.tw

J.-P. Sheu · W.-K. Hu · J.-C. Lin
Department of Computer Science and Information Engineering, National Central University, Jhongli 32001, Taiwan

1 Introduction

A wireless sensor network (WSN) consists of hundreds or thousands of low-cost sensors, which are capable of wireless communication and data processing. Collaborative execution among a large set of sensor nodes is required for applications of WSNs [5, 6]. Most of the applications need sensor nodes to be synchronized at the initial phase of the system such as objects tracking [18] and events detection [19]. Furthermore, time synchronization is essential for scheduling algorithm such as Time Division Multiple Access (TDMA) [17] which can be used to share the transmission medium in the time domain to eliminate transmission collisions and conserve energy. Therefore, time synchronization is an important issue of a sensor network as in all distributed systems. Many solutions exist for the traditional Internet and distributed systems [1–3, 11]. It is unsuited to implement the traditional time synchronization protocols on the sensor nodes even they have been largely successful in Internet. These Internet protocols assume the existence of a master clock, constantly connected, and consistent communication delays in data exchanges. Unfortunately, none of these assumptions is true in WSNs. Although lots of time synchronization protocols have been proposed in wireless ad-hoc networks [4, 12], they are infeasible for wireless sensor networks. The time synchronization requirements differ drastically in the context of sensor networks such as precision, computation complexity, and storage constraint.

As the increasing attention to the time synchronization problems in wireless sensor networks, numerous protocols suited to WSNs have been proposed [8–10, 13–16]. Time synchronization protocols can be categorized into three models. First is to figure out the event ordering. Many applications of sensor networks rely on event ordering to obtain useful information from sensed data. The clocks need

not be synchronized at the moment event happens. An approach for sensor network based on this kind of synchronization protocol was proposed in [7]. The second model is to maintain relative clocks. In this model, every node records the relative offset between its clock and the clock of any other node in the network. Reference Broadcast Synchronization (RBS) [8] is a protocol based on this model. The third one is that every node maintains a clock that is synchronized to a reference node. A global and unique time scale throughout the network is maintained. There are many protocols presented to solve the time synchronization problem in this model such as TPSN [16] and FTSP [13]. In this paper, we are interested in the scenario of the second and third models.

Time-stamping is an important operation of a sensor node and it is necessary for time synchronization protocol. By doing this operation at MAC layer, we can avoid the synchronization error brought by the packet delivery between the upper layers. But there still exist several delay times during transmission after time stamping. These delay times such as encoding, decoding, and propagation times are uncertain and affect synchronization error directly. Besides, clock drift is a main source of time synchronization error. Therefore, in this paper, we propose a Ratio-based Time Synchronization Protocol (RSP) to estimate the clock drift between two nodes to keep them synchronized for duration after once synchronizing. We implement our protocol and FTSP in the Berkeley MICAz platform to evaluate their performance. The experimental results show that the average time synchronization error between two nodes in our RSP protocol is less than that of FTSP. We also evaluate the time synchronization error of our protocol in the scenarios of 5-node and 18-node multi-hop topologies, and the resynchronization periods are 30-second and 300-second. The experimental results show that the average synchronization errors of all nodes run with our protocol are ranged within several micro-seconds.

The rest of this paper is organized as follows. In Sect. 2, we review the related work of time synchronization protocol. Then we describe our proposed protocol in Sect. 3. The evaluated performance of our time synchronization protocol is presented in Sect. 4. Finally, we offer our conclusion in Sect. 5.

2 Related work

Time synchronization has been one of the important research topics of wireless sensor networks in the past few years. Extensive researches have been conducted on how to transfer time or synchronized clocks that are distributed over wireless sensor networks. Before studying other existing time synchronization algorithms, we first discuss the main sources of time synchronization errors.

The adversaries of precise time synchronization in wireless networks include clock drift and uncertain delay times in packet transmission. The clock drift is the time offset between two different clocks at a fixed time period. Actually, the oscillator of each sensor node runs at the frequency which is slightly different from other's one. This phenomenon causes the clock to gradually diverge from each other. Besides, the clock frequency is unstable due to environmental conditions, such as temperature, humidity, and clock's aging. Note that, because of clock drift, even if the clocks of two nodes are synchronized at some time, they may not agree the same time in the future. In addition to clock drift, uncertain delay times in packet transmission also directly impact on the required precision of time synchronization. To realize these uncertain delay times, we will use the following decomposition of message delivery delays first introduced in [3] and later extended in [16].

The send time is spent to construct the message and issue the send request to the MAC layer on the sender. It includes kernel protocol processing and variable delays introduced by operating system, e.g. context switching and system calls overhead. This delay time is highly non-deterministic. The access time is incurred by waiting for an accessible channel for transmitting. This delay in WSN is varying from milliseconds up to seconds depending on the current network traffic and the MAC protocol in use. The transmission time is the time for a sensor node to transmit a packet at physical layer over a radio link. It depends on the length of the packet and speed of the equipped radio. The propagation time is the time taken by the packet to propagate over the wireless link from sender to receiver once it has left the sender. It is highly deterministic in WSN and can be negligible regarding others. The reception time is the time for a sensor node to receive the packet. It is the same as the transmission time. Finally the receive time is similar to the send time. It is the time to process the incoming message.

In [16], the authors present a method to completely eliminate the send, access, and receive times. Instead of time stamping the packet at the application layer, they time stamp the packet at the MAC layer. In other words, the packet is time stamped when the packet is about to be transmitted or received. Hence the above three delay times can be removed by MAC layer time stamping. Therefore, clock drift is left as the main source of time synchronization errors. In this paper, we only consider to eliminate the clock drift error existing among sensor nodes.

In the following, we review the existing algorithms that deal with time synchronizations in sensor networks. The authors in [8] proposed a Reference Broadcast Synchronization (RBS) protocol. The key idea of RBS is to use the broadcast nature of the wireless communication medium to reduce delays in the synchronization protocol. According to the broadcast nature of the wireless communication

medium, two receivers located within listening distance of the same sender will receive the same message at approximately the same time. The receivers within the communication range of a broadcast message sent by a reference node synchronize with one another, rather than with the reference node. This is accomplished by having the reference node broadcasts message to its neighbors. Recipients use the arrival time of the message as a point of reference for comparing their clocks. The recipients then exchange this information among themselves. In this way, they can compute the clock offset with each other. The time offset is the time difference between two sensor nodes.

The RBS scheme can be extended in two ways to achieve higher synchronization precision. One is to increase the number of reference messages broadcasted by the reference node and average the clock offsets from multiple reference messages. Instead of averaging the offsets, the other one is to perform a least squares linear regression in a sequence of m time offsets. This offers a fast and closed-form method for finding the best fit line over time. One reference message provides a synchronization point (local time of B, time offset) to receiver B for the least squares linear regression. After receiving m reference messages, m synchronization points are gathered. And then the best fit line can be found by these m synchronization points. By the line, the time offset of receiver B with respect to receiver A can be computed with local time of receiver B.

In [16] the authors proposed a time synchronization protocol for the wireless sensor networks, called Timing-sync Protocol for Sensor Networks (TPSN). The algorithm works in two phases: level discovery phase and synchronization phase. In the level discovery phase, a hierarchical tree structure is established. The reference node is assigned to level 0 initially, and every sensor node will be assigned a level during this level discovery phase. In the synchronization phase, a pair-wise synchronization is performed along the edges of this structure to establish a global time scale through the network. A sensor node belonging to level i synchronize to a sensor node belonging to level $i-1$. Each node is synchronized by exchanging two synchronization messages with its parent node. Eventually, all nodes in the network synchronize their times to the reference node. However, the TPSN did not handle the clock drift, which limits its performance.

The authors in [13] proposed a robust and low communication overhead synchronization protocol, called Flooding Time Synchronization Protocol (FTSP). The FTSP achieves time synchronization between a sender and possibly multiple receivers utilizing a single radio message time-stamped at the both sender and receiver sides. A single, dynamically elected node, called the reference node of the network, maintains the global time and all other nodes synchronize their clocks to the local time of the reference node. The nodes form an ad-hoc structure to transfer the global time from the reference node to all the nodes. The FTSP synchronizes the receivers to the time provided by the sender of the broadcast message. The broadcast message contains the sender time-stamp which is the global time. The receivers get the corresponding local time from the local clock when receiving the message. In this way, one broadcast message provides a synchronization point (global time, local time) to each of the receivers. The difference between the global and local time of a synchronization point estimates the time offset of the receiver. And the FTSP estimated the clock drift using linear regression through the synchronization points as suggested in RBS.

3 Ratio-based time synchronization protocol (RSP)

The purpose of the RSP is to perform wide time-synchronization of all nodes in the network with multi-hop topology. The RSP use two synchronization messages to synchronize the clock of the receiver with that of sender. By marking the time-stamp of the message in MAC layer, some delay time in handling the transmission can be eliminated and the accuracy is also enhanced. The RSP also can extend to multi-hop synchronization. One node in the network would be elected as a *synchronization root* by the leader election algorithm [13]. This root node will maintain the global time. All other nodes synchronize their clocks to that time of the root. The nodes in the wireless sensor network construct a tree structure and the root of this tree is the synchronization root. The global time of the root is flooding out to the nodes through the tree structure.

The local clock time of a sensor device is provided by the quartz oscillator inside itself. In the time t of the Coordinated Universal Time (UTC), we use $C_i(t)$ to present the local clock time of a sensor node i. The following is the transformation formula between t and $C_i(t)$:

$$C_i(t) = a_i(t) \times t + b_i \tag{1}$$

where $a_i(t)$ is the drift ratio and b_i is the offset of node i's clock at time t. The oscillator's frequency is not stable during working so that the value $a_i(t)$ is changed with time. And b_i for each node will also be different due to the time to switch on.

By (1), the local clock times of two sensor nodes i and j have the following relationship:

$$C_j(t) = \theta_{ij} \times C_i(t) + \phi_{ij}. \tag{2}$$

The parameters θ_{ij} and ϕ_{ij} represent the relative drift ratio and offset between the clocks of nodes i and j. As two nodes in the WSN want to execute the RSP, one of them would be a reference node and the other one is being synchronized to it. The reference node initiates the synchronization procedure by sending two continuous synchronization messages $SyncMsg_1$ and $SyncMsg_2$ with time-stamps T_1

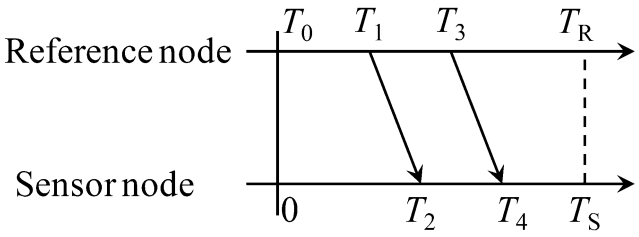


Fig. 1 Timing diagram for synchronization procedure of RSP

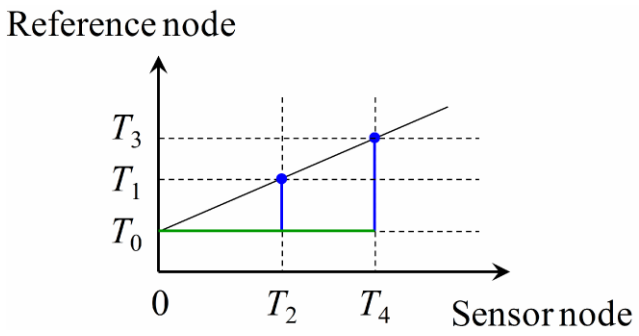


Fig. 2 Expression of linear interpolation with the four time-stamps

and T_3, respectively. All the neighboring nodes get the corresponding local time (T_2 and T_4) from its local clock when receiving the synchronization messages. Consequently, those nodes collect four time-stamps information as shown in Fig. 1. Then, each node can calculate the clock drift ratio between the reference node and itself from the four time-stamps, which is $\theta = (T_3 - T_1)/(T_4 - T_2)$.

According to the ratio, each node can estimate the local time of reference node in the following way.

$$\begin{aligned}
&(T_R - T_0) : (T_S - 0) = (T_3 - T_1) : (T_4 - T_2) \\
&\quad \Rightarrow \quad (T_R - T_0) \times (T_4 - T_2) = T_S \times (T_3 - T_1) \\
&\quad \Rightarrow \quad T_R = T_S \times \frac{T_3 - T_1}{T_4 - T_2} + T_0, \qquad (3)
\end{aligned}$$

where T_S represents the local time of sensor node and T_R represents the corresponding local time of the reference node. Additionally, T_0 ($= \phi$) is the initial offset between reference node and sensor node. In other words, the T_0 is the local time of reference node when the local time of senor node is 0. It can be calculated using linear interpolation with the four time-stamps as shown in Fig. 2. Thus, the T_0 can be derived as follows.

$$\begin{aligned}
&(T_4 - 0) : (T_2 - 0) = (T_3 - T_0) : (T_1 - T_0) \\
&\quad \Rightarrow \quad T_4 \times (T_1 - T_0) = T_2 \times (T_3 - T_0) \\
&\quad \Rightarrow \quad T_0 = \frac{T_1 \times T_4 - T_2 \times T_3}{T_4 - T_2}. \qquad (4)
\end{aligned}$$

Therefore, we can derive (5) from (3) and (4):

$$T_R = T_S \times \frac{T_3 - T_1}{T_4 - T_2} + \frac{T_1 \times T_4 - T_2 \times T_3}{T_4 - T_2}. \qquad (5)$$

By (5), each sensor node can estimate the local time of reference node, that is, the global time of the network.

In the above equation, we assume that the oscillators of the sender and receiver run in a stable frequency during the WSN working period. However, the oscillator frequency of each sensor node varies unpredictably due to various physical effects in practice. It results that there exists a time difference between the estimated local time of the reference node and its real local time. As time goes on, the time difference becomes more and more. In our RSP, the reference node will broadcast a synchronization message to the sensor nodes periodically. Let T_{send} denote as the received time-stamp of the reference node and T_{send} denote as the corresponding time-stamp of the sensor nodes. Then, each sensor node will replace T_3 and T_4 with T_{send} and T_{recv} to estimate the new local time of the reference node by (5). However, the time difference between $T_3 - T_1$ (or $T_4 - T_2$) will become more and more as time goes on.

In the following, we will consider the impact of time interval for time synchronization. Since the MCU of the sensor platform in general has limited calculating capacity, it would cause some error at the number behind the decimal point. Thus, the smaller synchronization time interval will have the larger estimation error in (5). Besides, the clock drift is unstable and changed with time. As using a large time interval, the relative drift ratio of two sensor nodes will become unreliable. Therefore, the time-stamps of (5) are needed to be refreshed if the time interval is larger than a threshold α. In our protocol, we will select two new pairs of (T_1, T_3) and (T_2, T_4) in (5) if the time difference between $T_3 - T_1$ (or $T_4 - T_2$) is larger than α. To replace the time-stamps of (5), each node will keep the latest k pairs of time-stamps T_{send} and T_{recv}. When a sensor node receives a new T_3 (T_{send}) sent from the reference node and finds that the time difference between $T_3 - T_1$ is larger than α, it will select a stored pair of time-stamps T_{send} and T_{recv} which are received before as the new T_1 and T_2. To reduce the numerical computation error for the MCU, the new T_1 and T_2 must satisfy the condition of $T_3 - T_1$ (or $T_4 - T_2$) larger than a threshold β.

For example, assume the synchronization message is sent by the reference node per 3 minutes, and α and β is set to 15 and 8 minutes, respectively. Each node records the latest $k = 5$ pairs of time-stamps T_{send} and T_{recv}. After the sixth time synchronizations, a sensor node will receive new time-stamp T_{send} and use it to replace T_3. The sensor node will find that $T_3 - T_1 = 18$ minutes $> \alpha$. It will choose a latest pair of T_{send} and T_{recv} as the new T_1 and T_2 such that $T_3 - T_1 > \beta$. In this case, we will select the fourth received T_{send} and its corresponding T_{recv} as the new T_1 and T_2, respectively, such that $T_3 - T_1$ (or $T_4 - T_2$) > 8 minutes.

In the following, we will extend our RSP from single-hop network to multi-hop one. Here, we assume each sensor node has a unique ID in the WSN. First, a root node is elected by the leader election algorithm in [13]. When a node is elected as the root in the WSN, it sets itself being a reference node. Then it broadcasts the time synchronization massages to the neighborhood nodes periodically. Any node received two continuous synchronization messages can use (5) to estimate the local time of the reference node. The local time of the reference node here is the global time. These neighborhood nodes then are turned into synchronized ones. Next, they also set themselves being reference nodes and do the same operations. The time synchronization messages transmitted by the synchronized nodes contain the time-stamps which are the global times estimated by them when being transmitting. Using these global times, other nodes received the messages can calculate the global time and become synchronized eventually.

In our proposed RSP, every synchronization message comprises five main fields which are *Send_ID*, *Seq_num, Time_stamp, Sync_root*, and *New_root*. *Send_ID* is the ID of the node which sends the synchronization message. *Seq_num* is a sequence number. As the root generates a new synchronization message to transmit, it increases this variable by one. Each node will keep k nearest received synchronization messages. *Time_stamp* is the global time estimated by the transmitter, and the receiver uses it to estimate the global time. *Sync_root* is the root ID which is reserved by the synchronized nodes. *New_root* is used to announce that a new root node is elected. In order to keep the network nodes synchronize to the same root, each sensor node has a local variable, *My_root_ID*, referred to the current reference root's ID of the network. As receiving a synchronization message whose *Sync_root* field is smaller than its *My_root_ID*, the node will uses the new root's global time-stamps to calculate its clock drift ratio θ and offset ϕ.

Our RSP is a tree-based synchronization scheme which is activated from the root node. As soon as the root is elected by the leader election algorithm, it will start the time synchronization procedure. Firstly, the root starts the time synchronization by sending the messages which contain the time-stamp information and sender ID. The nodes within the root's radio range receive the synchronization packets. Then they will correct their local clocks and become synchronized nodes. All synchronized nodes broadcast the new synchronization messages like the root. All non-synchronized nodes receive the synchronization messages will do the same actions as the nodes receiving message from the root. The above procedures will repeat until each node complete the same work. Finally, the network forms a virtual synchronization tree structure. A node will discard the redundancy messages sent from its neighbors. Finally, each node will synchronize with its parent node until the tree is broken due to the dead root.

A summary of our protocol is presented as follows:

Algorithm 1: Ratio-Based Time Synchronization

Input:

msg; /* time synchronization message */

α, β; /* thresholds for time synchronization */

$T = \{(T_x, T_y) | (T_x, T_y)$: the recent received k pairs of time-stamps $\}$;

Output:

θ, ϕ; /* the relative drift ratio and offset to the synchronization root */

Main()

{ $My\_root\_ID = My\_ID$; /* declare myself as the synchronization root */

Broadcast time synchronization message *msg* periodically;

Loop:

if (Receive any synchronization message *msg*)

{**if** (*My_root_ID* > *msg.Sync_root*) /* Check if a more suitable root exists. */

{**if** (*My_root_ID* == *My_ID*) stop the broadcast of time synchronization message *msg*;

$T_1 = msg.Time\_stamp$; /* the transmitter's clock time */

$T_2 = My\_local\_timestamp$; /* the receiver's clock time */

$My\_root\_ID = msg.Sync\_root$;

$My\_parent = msg.Send\_ID$;

$My\_sync\_seq = msg.Seq\_num$;

$T \leftarrow \Phi$; $T \leftarrow T \cup (T_1, T_2)$; }

else if (*My_Parent* == *msg.Send_ID* and *My_sync_seq* < *msg.Seq_num*)

{$T_3 = msg.Time\_stamp$; $T_4 = My\_local\_timestamp$;

$My\_sync\_seq = msg.Seq\_num$;

```
    if (T3 − T4 > α)
      {Find a timestamp pair (Tx, Ty) from T such that T3 − Tx > β;
       T1 = Tx; T2 = Ty; }
       θ = (T3 − T1)/(T4 − T2);
       φ = (T1 × T4 − T2 × T3)/(T4 − T2)
       T ← T ∪ (T3, T4);
       Broadcast a new synchronization message msg;}
      else
       Discard msg; }
    go to Loop;
}
```

Each node in the network may lost synchronization messages due to the tree structure is broken. When a node cannot receive the synchronization messages in the expected period time from its parent node, it will actively send a synchronization request back to its parent node. If it cannot get any response from the parent node in *Max_Req* times of the request, the node broadcasts a request to all its neighbors to ask a new parent node. The synchronized neighboring nodes which receive the request will send the synchronization message backward to it. When the node receives the first synchronization message and confirms its validity, it will record the sender as its new parent node. All recorded time-stamps sent from the old parent node are dropped. The node will use the time-stamps sent from the new parent to follow the time synchronization.

Here, we consider the root crashes or shuts down due to the exhausted power. If a node detects that its root parent crashed, it will become the new root node and starts the new time synchronization procedure. When a node declares itself as the new root, it will start a short synchronization period. In such situation, the interval of the first two synchronization messages is shorter than the regular one. It will promote the non-synchronized nodes to be synchronized as soon as possible.

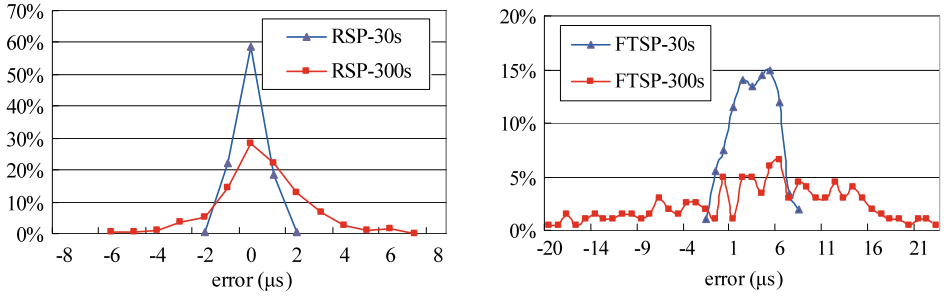
4 Experiments

In order to test and verify the performance of our RSP algorithm, we implement RSP and FTSP on the MICAz platform based on TinyOS. The FTSP protocol was designed by Vanderbilt University [13]. Based on our best knowledge, FTSP has the smallest time synchronization error in WSNs. Its source code program for TinyOS is public. There is one 32-bit system time in each MICAz and it will increase every 1 μs. In our experiments, all nodes need to report their current global time at the same time. To achieve this job, we add another two nodes which do not attend the time synchronization procedure. One is being a query node and responses for broadcasting a query message periodically; the other one is a sink which collects the packets sent by the synchronized nodes. All experiment topologies are set forcibly in software, and hence the query node and the sink can directly communicate with all nodes in the experiment scenarios. Thus, just one-hop transmission is used for query procedure and this can effectively reduce the additional synchronization errors during packet transmission. After the network start to synchronization procedures, the query node will broadcast a query packet periodically. Each synchronized node receives this packet at the same time. They will immediately access their global time as receiving the packet and then send their global time to the sink. The query intervals of the following experiments are all set to 10 seconds.

We first observed the synchronization errors of RSP and FTSP between two nodes. Each experimental result was the collection of 200 query data. In Fig. 3(a), we show the synchronization errors between two nodes of our protocol. In the 30-second re-synchronization case, the average absolute synchronization error is 0.42 μs and the synchronization error at 0 μs is 58%. All synchronization errors are between 2 μs and −2 μs. In the 300-second re-synchronization case, the synchronization error at 0 μs becomes 28% and the average absolute synchronization error increases to 1.73 μs. The error range is distributed between 7 μs and −7 μs.

The experimental results of FTSP are shown in Fig. 3(b). In the 30-second re-synchronization case, the synchronization error at 0 μs just takes 7.5% and the average absolute synchronization error is 3.33 μs. All synchronization errors are distributed between 8 μs and −2 μs. In the 300-second re-synchronization period, the synchronization error at 0 μs becomes about 5% and the average absolute synchronization error increases to 8.47 μs. The error range is distributed between 23 μs and −20 μs. According to the above results, the RSP outperforms the FTSP no matter how long of the time re-synchronization period was set.

Fig. 3 The distribution of the errors in single-hop



(a) RSP (b) FTSP

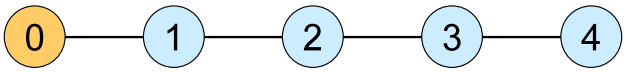
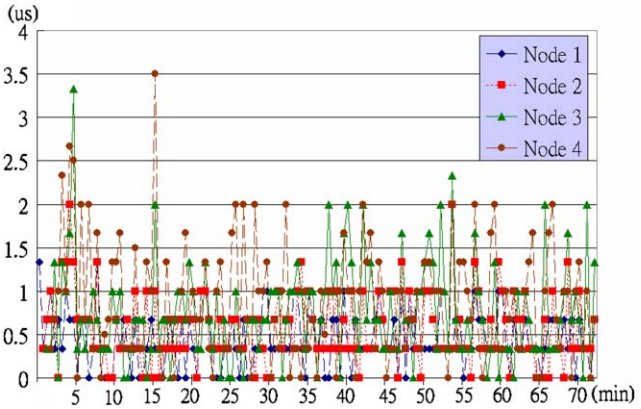


Fig. 4 The linear topology for multi-hop

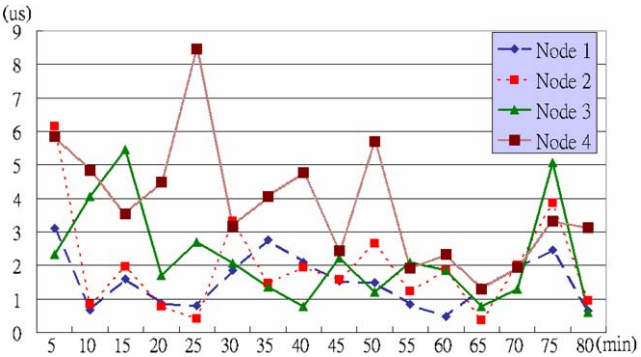
The second experiment was designed to observe the individual and cumulative synchronization errors as adding a hop for RSP. A 5-node topology for this experiment scenario is shown in Fig. 4. Each node can only communicate with its neighbors. The node with ID 0 is the reference one. The period of the time synchronization was set 30 seconds and 300 seconds, respectively. It totally runs one hour for this experiment.

In Fig. 5, we show the synchronization errors of multi-hop with linear topology. The x-axis represents the experiment time and the y-axis represents the average error within one re-synchronization period. For example, if the period of the time re-synchronization was set to 300 seconds, the data point in the result graph represented the average value of all collected errors during the last 300-second period. And all errors are compared with the reference node. For 30 seconds period, all four nodes' errors are under 4 μs as shown in Fig. 5(a). As the period is increased to 300 seconds, the maximum error becomes 8.5 μs as shown in Fig. 5(b). It is shown that our RSP still performs well as the period was enlarged. The maximum error of one node is the maximum value among all collected errors from it and the average error of one node is the average one from it. As re-synchronizing per 30 seconds, the average of node 1 and node 4 were 0.44 μs and 0.88 μs respectively; the maximum errors over all nodes were between 2 μs and 4 μs. When synchronizing through one new node, the error increased less than 1 μs. In the 300-second period of time synchronization, the average errors of all nodes during the whole experiment become larger and are between 1.53 μs to 3.83 μs. The overall maximum error happens at the node 4 and the value is 16 μs. From the above results, if the node is far from the reference node, the synchronization error would increase due to the more transmission hops.

The following scenarios include turning off some nodes, changing the root node, and adding new nodes into the network. This experiment was divided into several time durations. Each one had different network states. By the above actions, the topology of the network is changed as time goes on. It simulated the situations of the real WSN during operating. The general network topology is shown in Fig. 6. There are total 18 MICAz nodes deployed in a grid mode and each one has an ID number over it in the figure. Each node can only communicate with its neighbors which are connected by the lines in the figure. We use two time synchronization periods, 30 seconds and 300 seconds in this topology. If no synchronization message is received after six



(a) Re-synchronization per 30 seconds



(b) Re-synchronization per 300 seconds

Fig. 5 The experiment results of 5 nodes with linear topology

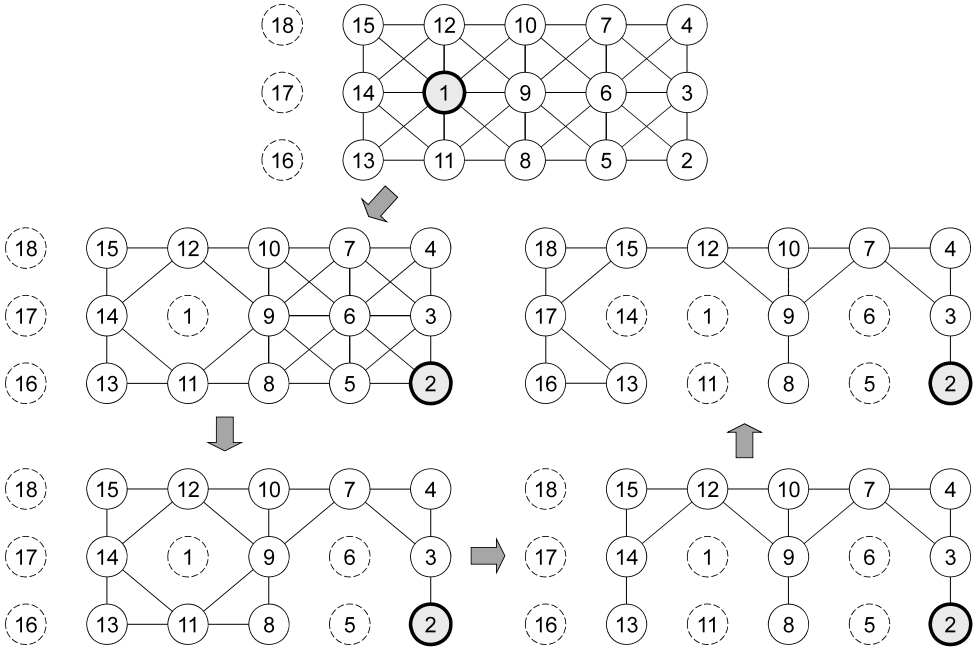


Fig. 6 The multi-hop scenario. The nodes drawn with *dash line* are turned off. The nodes drawn with *solid line* are active. The nodes drawn with *bold solid line* are the roots

time requests, each node would broadcast a request to all its neighbors to ask a new parent node.

The experiment took about 5 hours and each hour had a different topology state. Each individual state is as the following. At the first state (from *Start* to $t1$), nodes with numbers 16, 17, and 18 were turned off and other nodes were turned on. The node with number 1 (currently the smallest ID number) was the reference of the network. The longest hop count from the reference to other nodes is 3. At the beginning of the next state (from $t1$ to $t2$), node with number 1 was turned off and the node with number 2 became the new reference node. The longest hop count is increased to 4. At the third state (from $t2$ to $t3$), nodes with numbers 5 and 6 were turned off. The longest hop count becomes 5. At the forth state (from $t3$ to $t4$), one node with number 11 was turned off and the longest hop count is increased to 6. At the final state (from $t4$ to *End*), we turned off the node with number 14 and turned on the nodes with numbers 16, 17, and 18. The longest hop count from the reference node is increased to 7. Each state was kept run one hour.

For the above network topology, the synchronization errors graph is shown in Fig. 7. We now firstly concentrate upon the 300-second re-synchronization period. As at the initial stage, all nodes were manually switched on, each node's clock started at different time. Then for the competition of the root among nodes, there is a short period that all nodes are unsynchronized at the beginning of the experiment. Thus at the initial stage, no error results are shown in the figure. In the first few minutes, the average errors growing as the time goes on. It is because the elected root completes the first time synchronization by using a short synchronization interval (30 seconds). By such an interval, it

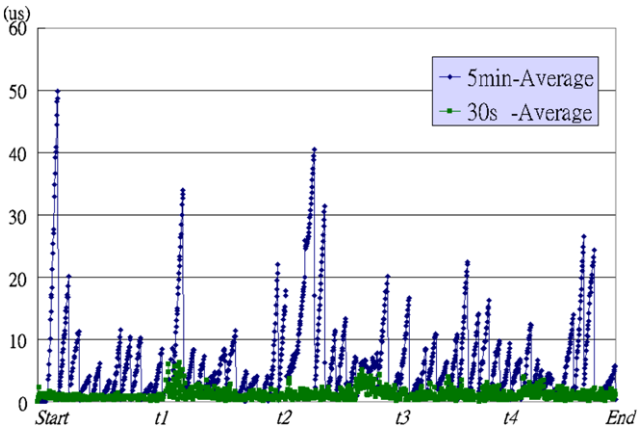


Fig. 7 The synchronization errors of the scenarios in Fig. 4

cannot fully represent the clock drift within the nodes and the function to transfer to the global time could not be estimated accurately for a long period. The average errors of synchronized nodes then increased gradually before the next re-synchronization procedure. When the new synchronization packet was received, the global time was estimated by the new time-stamps. The average errors were soon reduced to quite small. By querying per 10 seconds, we can obviously see the change of the average errors. The average errors was quite small while each time completing a new round of re-synchronization process.

At the second topology (time $t1$), the elected root was forced to be turned off. All nodes detected that no new synchronization packet at the expected time period. Later, they also did not get any response while actively querying their parent nodes for the synchronization message. At this time,

Table 1 The average and maximum errors for the five multi-hop scenarios

Error (μs)	Re-synchronize every 30 s		Re-synchronize every 5 min	
	Avg.	Max.	Avg.	Max.
start $\sim t1$	0.7	9	4.19	55
$t1 \sim t2$	1.16	8	4.55	38
$t2 \sim t3$	1.35	9	5.43	29
$t3 \sim t4$	1.13	9	5.77	40
$t4 \sim$ end	1.01	8	7.54	37

they would start competing to be the new root of whole network by executing the root election algorithm. During the competing period, all nodes still were synchronized by the old synchronization messages. But it was apparent that the average errors in the figure became larger and larger for a long time before the new root was elected. While the new root was elected and the new tree structure was built, the new synchronization procedure started operating. The average error decreased to quite small value immediately.

At the third topology (time $t2$), some nodes near the root were turned off. The little influence happened at the network. The longest hop of the network was increased and the average errors were larger than those at the normal situations. It is because the nodes near the shutdown nodes need to find their new parent nodes to be synchronized again. Therefore, the average errors were kept high before all nodes were synchronized once more. After two rounds of the synchronization procedure, the errors would be immediately reduced to the normal level.

At the next topology (time $t3$), just one node was turned off. The effect for this change was not so obvious. All nodes' average errors kept in the normal range. Subsequently when the new nodes were joined to the network (time $t4$), the longest hop count from the root was 7. It was the longest distance from the root during the whole experiment. The new nodes were synchronized with the network after receiving the synchronization packets. So after being synchronized, their estimated global times were collected to calculate the average errors. The average errors stably varied at the normal range and no obvious effect came into existence.

Figure 7 also presents the other experiment results with the 30-second synchronization period. At that period, the averages errors among the whole experiment were under 10 μs. When the root was turned off, the errors became a little large. It got pretty low synchronization errors as the synchronization period was shorter. The figure highlights the differences between the two synchronization periods.

We also calculated the average and maximum errors for the five periods of this experiment, respectively. Table 1 lists the values for the 30-second and 300-second re-synchronization experiments. The results reflected in Table 1 indicated that the interval of the re-synchronization period was significantly associated with the synchronization accuracy. The additions on the hop count from the root to the edge node in the 30-second re-synchronization period were no strong relationship for the synchronization errors. But it rose in the 300-second one since the average error is increased from 4.19 μs to 7.54 μs after adding the longest hop count. The maximum errors of the five periods in the 30-second re-synchronization period almost were the same. It was no high correlative with the change of the network topology. The overall maximum error, 55 μs, in the 300-second re-synchronization period appeared in the first experiment period. It was caused by the root which did the first time synchronization by a short synchronization interval. But after that, all synchronized nodes never got such high errors.

5 Conclusion

This paper proposes a Ratio-based Time Synchronization Protocol (RSP) in wireless sensor networks. By way of the periodical synchronization messages and the uncomplicated formulas, the presented RSP keeps the network nodes synchronized. The protocol provides that all kinds of applications or systems can operate with it and will not be affected during working. As soon as some applications require the accuracy of the time synchronization, the RSP can reach the requirements by do related works. On the other way, if the network applications are focused on the energy saving, it also can be made.

Our RSP maintains the tree structure to keep the network nodes synchronized effectively. With co-operating the periodical time-stamp packet transmission, it can be soon to recover the non-synchronized situations. The condition of failed root could be found in a specified time. Then the new root is elected from them to start the new round of time synchronization. Also some node losing synchronization with its reference node can retrieve in synchronized state by choosing one of the neighbor nodes. By these characteristics of RSP, all the terrible conditions against the time synchronization can be reduced to minimum. Therefore, the RSP can provide the good robustness and increase the efficiency for time synchronization.

We implement our proposed RSP and FTSP in the MICAz platform with TinyOS and evaluate their effectiveness. According to the experimental results, the average time synchronization error between two nodes in our RSP protocol is less than that of FTSP. Next, two scenarios of multi-hop topologies are also set up to experiment our RSP protocol. In the chain topology, the performance of 30-second resynchronization period is better than that 300-second one. Then the RSP is tested on a larger network topology. The presented RSP still accomplishes great accuracy and good robustness by simulating the variance of the network situation in the real experiments. The results show the proposed RSP can be applied to the majority of the WSN applications.

Acknowledgements This work was supported by the National Science Council of Republic of China under grants NSC 96-2221-E-008-005 and NSC 95-2752-E-007-003-PAE.

References

1. Liao, C., Maronosi, M., & Clark, D. (1999). Experience with an adaptive globally-synchronizing clock algorithm. In *Proceedings of the 11th annual ACM symposium on parallel algorithms and architectures* (pp. 106–114).
2. Dai, H., & Han, R. (2004). TSync: a lightweight bidirectional time synchronization service for wireless sensor networks. In *Newsletters of ACM SIGMOBILE mobile computing and communications review* (pp. 125–139).
3. Kopetz, H., & Ochsenreiter, W. (1987). Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers*, *36*(8), 933–940.
4. Sheu, J.-P., Chao, C.-M., & Sun, C.-W. (2004). A clock synchronization algorithm for multi-hop wireless ad hoc networks. In *Proceedings of the 24th international conference on distributed computing systems* (pp. 574–581).
5. Sheu, J.-P., Hsu, C.-S., & Li, J.-M. (2006). A distributed location estimating algorithm for wireless sensor networks. In *Proceedings of international conference on sensor networks, ubiquitous, and trustworthy computing (SUTC 2006)* (pp. 218–225).
6. Sheu, J.-P., & Ding, M.-L. (2007). Routing with hexagonal virtual coordinates in wireless sensor networks. In *Proceedings of the IEEE wireless communications and networking conference (WCNC)* (pp. 2929–2934).
7. Elson, J., & Estrin, D. (2001). Time synchronization for wireless sensor networks. In *Proceedings of the 15th international parallel and distributed processing symposium* (pp. 1965–1970).
8. Elson, J., Girod, L., & Estrin, D. (2002). Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the 5th symposium on operating systems design and implementation* (pp. 147–163).
9. Elson, J., & Römer, K. (2002). Wireless sensor networks: a new regime for time synchronization. In *Proceedings of the 1st workshop on hot topics in networks* (pp. 149–154).
10. Greunen, J. V., & Rabaey, J. (2003). Lightweight time synchronization for sensor networks. In *Proceedings of the 2nd ACM international conference on wireless sensor networks and applications* (pp. 11–19).
11. Arvind, K. (1994). Probabilistic clock synchronization in distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, *5*, 474–487.
12. Römer, K. (2001). Time synchronization in ad hoc networks. In *Proceedings of the 2nd ACM international symposium on mobile ad hoc networking and computing* (pp. 173–182).
13. Maróti, M., Kusy, B., Simon, G., & Lédeczi, Á. (2004). The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on embedded networked sensor systems* (pp. 39–49).
14. Sichitiu, M. L., & Veerarittiphan, C. (2003). Simple, accurate time synchronization for wireless sensor networks. In *Proceedings of wireless communications and networking* (pp. 1266–1273).
15. Su, P. (2003). *Delay measurement time synchronization for wireless sensor networks*. TechIntel Research Berkeley Lab, IRB-TR-03-013.
16. Ganeriwal, S., Kumar, R., & Srivastava, M. (2003). Timing-sync protocol for sensor networks. In *Proceedings of the 1st international conference on embedded networked sensor systems* (pp. 138–149).
17. Rappaport, T. S. (1999). *Wireless communication principles & practice*. New Jersey: Prentice Hall.
18. Xu, Y., Winter, J., & Lee, W.-C. (2004). Prediction-based strategies for energy saving in object tracking sensor networks. In *Proceedings of the 2004 IEEE international conference on mobile data management* (pp. 346–357).
19. Cui, X., Hardin, T., Ragade, R. K., & Elmaghraby, A. S. (2004). A Swarm-based fuzzy logic control mobile sensor network for hazardous contaminants localization. In *Proceedings of the 2004 IEEE international conference on mobile ad-hoc and sensor systems* (pp. 194–203).

Jang-Ping Sheu received the B.S. degree in computer science from Tamkang University, Taiwan, Republic of China, in 1981, and the M.S. and Ph.D. degrees in computer science from National Tsing Hua University, Taiwan, Republic of China, in 1983 and 1987, respectively.

He is currently a Chair Professor of the Department of Computer Science, National Tsing Hua University. He was a Chair of Department of Computer Science and Information Engineering, National Central University from 1997 to 1999. He was a Director of Computer Center, National Central University from 2003 to 2006. His current research interests include wireless communications and mobile computing. He was an associate editor of Journal of the Chinese Institute of Electrical Engineering, Journal of Information Science and Engineering, Journal of the Chinese Institute of Engineers, and Journal of Internet Technology. He is an associate editor of the IEEE Transactions on Parallel and Distributed Systems, International Journal of Ad Hoc and Ubiquitous Computing, and International Journal of Sensor Networks.

He received the Distinguished Research Awards of the National Science Council of the Republic of China in 1993–1994, 1995–1996, and 1997–1998. He received the Distinguished Engineering Professor Award of the Chinese Institute of Engineers in 2003. He received the certificate of Distinguished Professorship, National Central University in 2005. He received the K.-T. Li Research Breakthrough Award of the Institute of Information and Computing Machinery. Dr. Sheu is a senior member of the IEEE, a member of the ACM, and Phi Tau Phi Society.

Wei-Kai Hu received the B.S. degree in Computer Science and Information Engineering from Tunghai University, Taichung, Taiwan, in 2001. He now studies for master and PhD degree in Department of Computer Science and Information Engineering of National Central University, meanwhile. His current research interests include wireless sensor networks, Ad Hoc networks.

Jen-Chiao Lin received the Master degree in Computer Sciences and Information Engineering from the National Central University, Taiwan, R.O.C. Her research interests include wireless sensor networks, time synchronization, and localization.