Enhanced Tree Construction Algorithm for Timing-Sync Protocol for Sensor Networks

Shi Kyu Bae

***Abstract*—Timing-sync Protocol for Sensor Networks (TPSN), the representative of time synchronization protocol for wireless sensor networks (WSN), was developed to provide higher synchronization accuracy and energy efficiency. So, TPSN's approach has been referenced by so many other WSN synchronization schemes till now. Although authors of TPSN realized that the network-wide performance of TPSN depends on the efficiency of the hierarchical structure, they used simple tree construction algorithm because of its simplicity and lower algorithmic overhead. In this paper, I propose a new tree construction algorithm for TPSN, which has low complexity and gives better efficiency to TPSN's synchronization process. The proposed algorithm's performance has been evaluated by simulation. The results are shown to be better than the original algorithm used in TPSN.**

***Index Terms*—Time synchronization, TPSN, wireless sensor network, tree construction.**

I. Introduction

Wireless Sensor Networks (WSN), which utilizes wireless implementation of sensor nodes, can be applicable to so many areas such as environmental monitoring, military surveillance, and so on. Many WSN applications consider local clocks at each node that need to be synchronized to a common reference time. Thus, Time Synchronization is one of the important issues in WSN, as well as other computer network area, where clock offset and drift cause all nodes to be asynchrony problem with each other. Such examples that time synchronization plays a crucial role are data fusion, data aggregation, duty cycling, transmission scheduling, localization, security, tracking etc.

Several works have been proposed to cope with time synchronization for WSN. Timing-sync Protocol for Sensor Networks (TPSN) [1] was developed which is based upon traditional Network Time Protocol (NTP) [2].

TPSN requires the hierarchical topology which is created before synchronization, (which is called level discovery in TPSN). The tree construction algorithm used in TPSN is simple, and has room to be enhanced, which the authors already considered.

In this paper, I propose a new tree construction algorithm for TPSN, which has low complexity and makes TPSN's synchronization process efficient.

The remainder of this paper is organized as follows. After surveying related existing works for time synchronization in Section II, TPSN is reviewed for enhancement in Section III. In Section IV, a new tree construction algorithm is proposed. Simulations for evaluating the proposed algorithm are performed in Section V. This paper ends with some concluding remarks in Section VI.

Manuscript received September 24, 2014; revised January 10, 2015.

Shi Kyu Bae is with DongYang University, South Korea (e-mail: skbae@dyu.ac.kr).

II. Related Works

Several works have been proposed to resolve time synchronization for WSN, especially. TPSN is the most popular protocol among them.

Reference Broadcasts Synchronization (RBS) [3], which was developed in 2002, lets a sender broadcast beacon [s] for receivers' reference, and receivers except the sender participate in synchronization by exchanging their observation after recording the time that the beacon was received. RBS increases the accuracy by eliminating sender side's delay uncertainty. Even though good synchronization accuracy, RBS cannot transmit exact global reference time efficiently (i.e. focusing on relative clock synchronization).

TPSN [1], which was developed in 2003, operates basically like NTP. Both TPSN and NTP measure round trip delay and estimate clock offset between two nodes. Additionally, TPSN uses timestamps at Medium Access Control (MAC) layer to improve delay measurement accuracy, other than NTP.

In Flooding Time Synchronization Protocol (FTSP) [4] developed in 2004, the authors proposed to use broadcast, not unicast unlikely TPSN, and uses timestamps at MAC layer for the similar reason as TPSN.

These Time Synchronization schemes have been classified into two categories in terms of message flow; *Sender*-receiver (SR) and *Receiver-receiver* (RR) [5]. *SR* type, which includes TPSN and FTSP, indicates that one node sends a message, whereas the others receive it. In RR approach, receivers mainly participate in synchronization, rather than the sender, which is the case of RBS. In wireless networks synchronization, using broadcast (i.e. RBS and FTSP) is more advantageous than one using unicast like TPSN.

Meanwhile, synchronization schemes have two different approaches according to different goal to synchronize; *Absolute* and *Relative* synchronization, which are similarly classified into *internal* synchronization versus *external* synchronization in [5]. Absolute (or external) synchronization is referenced to global reference time. In Relative (or internal) synchronization, a global reference time base is not available or not necessary. So the protocol attempts and focuses on minimizing clock offsets among nodes.

In the light of absolute or relative clock synchronization, *SR* type schemes synchronize with either absolute or relative clock, whereas *RR* type schemes focus on only relative clock.

III. Review of TPSN

TPSN is composed of two phases; that is, *level discovery* and *synchronization phase*. *Level discovery* phase creates the hierarchical topology of the network in the form of Spanning Tree starting from the root node, as shown in Fig. 1. Each node is assigned a level (i.e. a group of nodes with same depth from the root node), where the root node is level zero.

In synchronization phase, each node at a higher level (including root node) initiates synchronization to all children nodes by broadcasting, and each child node synchronizes with its parent node by two-way exchange of messages. After all higher level's nodes from the root node finish this basic synchronization steps to the last lowest level's nodes sequence by sequence, network-wide synchronization completes.

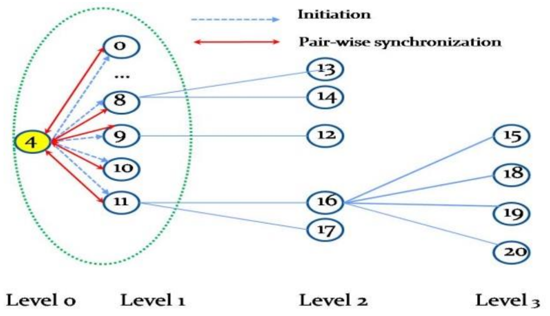

Fig. 1. A typical tree used in TPSN.

Suppose $n(T)$ is the number of nodes of an entire network. The hierarchical tree of TPSN is constructed by simple flooding-based method, and transmissions as many as $n(T)$ times are required.

Here, let me define notations which will be used in this paper, as shown in Table I below.

TABLE I: Definition of Notations Used in This Paper

notation	Meaning
T_i	Tree
n_i	Node in a tree
$h(T)$	Height of a tree
$l(n_i)$	Level of a node
$p(n_i)$	Parent of a node
$P(T)$	Set of parents nodes at a tree
$n(P)$	The number of set $P(T)$
$n(T)$	The number of nodes in a tree
$SOL(T)$	Sum of levels for all nodes in a tree
M	Total number of message transmissions per synchronization round
R	The radius of broadcast range, which is determined by transmission power at node.

$n(P)$ ranges from 1 to $n(T)$. Fig. 2 shows special cases of the tree for TPSN; one-hop topology (Fig. 2 (a)) and the longest multi-hop topology (Fig. 2 (b)).

One initiation message per parent node and one round-trip message exchange per child (or link between the parent and one of its children) are required in synchronization phase of TPSN as shown in (Fig. 1 or Fig. 2). Thus, the total number of message transmissions per round can be calculated below.

$$M = n(P) + 2(n(T) - 1) = 2 \times n(T) + n(P) - 2 \quad (1)$$

The power consumption of a WSN is known to be proportional to the size of data transmitted. An author of [6] showed that the total energy consumption of a network for WSN applications can be evaluated in terms of the total size of the transmission messages generated in the entire network, if transmission range is determined and fixed at the transmitter in a uniformly deployed static WSN (i.e., constant node density).

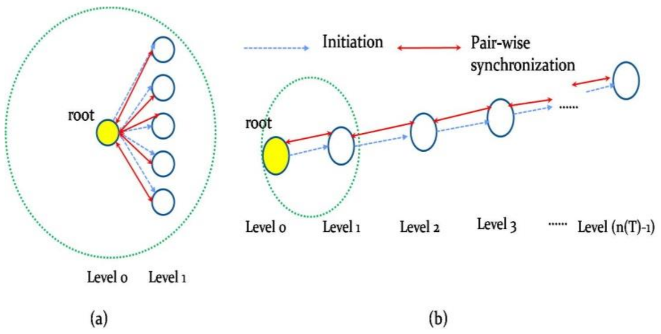

Fig. 2. Special cases of trees used in TPSN.

IV. Proposed Scheme

A. Characteristic of TPSN

Although TPSN is very useful and popular time synchronization protocol, it has some room for improvement.

Authors of TPSN realized that the network-wide performance of TPSN depends on the efficiency of the hierarchical structure. However, they used simple flooding for level discovery because of its simplicity and lower algorithmic overhead.

In TPSN, as number of level in a tree increases, clock offset between a node and the root node which has global reference time also increases [1]. As a result, synchronization errors through the network are getting bigger. Therefore, the smaller the depth of a tree is, the better synchronization accuracy is.

In Fig. 1, for instance, nodes at the level 2(node #12 or #16) are less accurate than nodes at the level 1(node #0 or node #11). Nodes at the lowest level (node #15 or node #20) are worst in terms of synchronization accuracy.

There have been some tries to improve TPSN by reducing a tree depth [7]-[9]. The authors of [7] used cluster hierarchy with a level depth fixed. And the authors of [8] proposed that a child node can select a parent with the lowest level from several nodes.

If a new tree construction algorithm, which achieves better performance of synchronization and has the almost same overhead as one used in TPSN, is available, it will be useful.

B. Tree Construction in TPSN

To make a tree, TPSN initiates tree construction by sending a level-discovery packet at root node. Root node sets its level to zero and broadcasts a packet with {node-id, level #}. Level is the hop distance from root node. So the level number is set to 0 at root node, and increases as the step makes progress.

In the case of root node, parent's node-id is "-1" because of no parent. When each node receives the packet, it sets its parent node with the received node-id, and its level with 1 more than the value that received. After that, a node broadcasts a packet again in the same previous way after random waiting-time. The sent packet makes other nodes be the next children, and makes parent-child relationship. This process is repeated again until all nodes in a network are set. The node having no child (i.e. lead nodes) will not initiate synchronization in the synchronization phase.

In original TPSN level discovery, the next parent node[s] at lower layer will be determined by start-up time of the parent node, which is randomly selected. Random wait time to avoid collision among adjacent nodes causes not only increasing tree construction time, but also producing different trees whenever level discovery operates.

Assume that no wait time is used for each node in TPSN level discovery. The next parent nodes at lower level are mainly determined by distance from the node of higher level. In other words, nodes operate in a way of the *'first-come-first-started'*, similar to*' first-come-first-served'*. For example, if node #1 is root node in the network topology in Fig. 3, nearer nodes from the root node (node #0, node #2, or node #4) will be the next parent nodes in advance, rather than farther nodes (node #5, node #3, or node #7).

If farther nodes become the next parent nodes, instead of nearer nodes, total number of levels for the entire nodes or depth of the tree will be considered to be reduced. For example, node #7 becomes the next parent nodes instead of node #4, as shown in Fig. 3 (b), the depth of the tree will be smaller, which means the end nodes such as node #19 or node #20 will has smaller level value (i.e. higher level than the former case).

C. Proposed Tree Construction Algorithm

If a tree for TPSN has a reduced number of levels for all nodes in a network, synchronization accuracy through the network will be improved, which was discussed in [7], and [8]. So, I propose a new tree construction algorithm for this purpose.

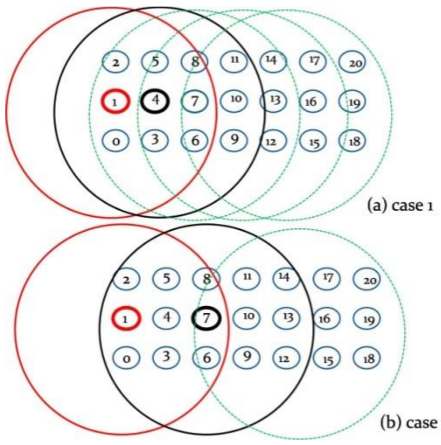

Fig. 3. Level discovery progress: (a) TPSN's case with zero wait time, (b) other method.

In my tree construction algorithm, level discovery packet is initiated from root node as the same way as TPSN level discovery phase. However, other nodes except root node do not start another broadcast immediately or after random wait time. Instead, each node postpones broadcasting for wait time which is calculated considering the distance between the parent node (that is, the sent node) and the node itself.

The wait time (W) is determined inversely proportionally to the distance between the sent node and the node itself as follows.

Let R is the radius of broadcast range which is determined by transmission power at node. And D is the distance between two nodes and k is constant.

$$W = k \times \frac{R}{D} \text{ (s)} \tag{2}$$

Distance between the sent node and the node itself can be measured by two ways;

First, if each node knows its position at the time of deployment, or when being notified by the base station, it can notify its position to other nodes by transmitting a packet. The receiver node can calculate the distance between the two nodes with their coordinates.

Secondly, if a receiver node can detect the strength of the received signal from the sent node, it can also calculate the distance between the sent node and the node itself.

Fig. 4 shows a pseudo code of the proposed algorithm.

```
<proposed tree construction algorithm>
Radius : radius of broadcast range
if (root node)
   Broadcast a discovery-level packet (node-id, level#)
For all nodes except root node {
  If (packet received && firstly received) {
     Set level=parent node's level+1;
     Distance = measure distance between parent node and node itself
     **Wait (k*Radius/distance) seconds**;
     Broadcast additional discovery-level packet
  }
```

Fig. 4. Pseudo code of the proposed algorithm.

V. Simulation Test

The proposed tree construction algorithm was tested by simulation with Network Simulator, NS2- 2.34 [10].

The simulation has been performed with 36 nodes in the topology shown in Fig. 5. Distance among adjacent nodes in x coordinate is 50m, and distance in y coordinate is 100m uniformly, and communication region diameter is 550m. *Node 0* is set to be root node, and *Red circle* represents the broadcast range. TDMA MAC and AODV routing protocols are used in this test.

In original TPSN level discovery, the next parent nodes at a lower layer are determined by random start-up time to avoid collision, which makes different trees whenever test operates.

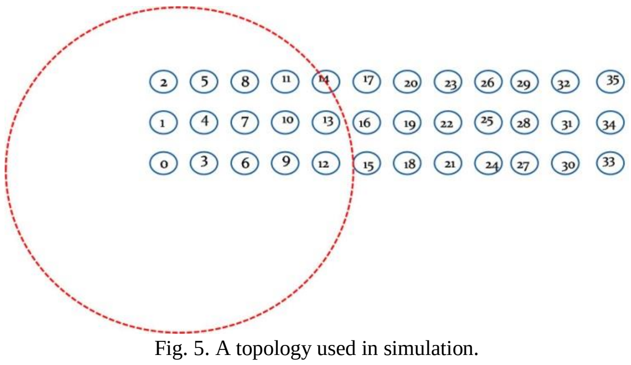

Fig. 5. A topology used in simulation.

TPSN tree construction was performed in condition of uniform random wait time from 0.0(s) to 1.0(s) several times.

As shown in Table II, SOL(T) and $n(P)$ range from 62 to 70, and from 4 to 12, respectively. And $h(T)$ was 3 or 4.

TABLE II: TREES CREATED TPSN ALGORITHM

Tree types	a	b	c	d	e	f	g	h
SOL(T)	62	62	63	64	65	66	67	70
$h(T)$	3	3	4	3	3	3	3	4
$n(P)$	10	12	8	9	4	6	7	8

In case of zero wait time, the following tree was created; $SOL(T)$=62, $h(T)$=3, $n(P)$=21.

When wait time varies from 0.0(s) to 0.1(s), the result is the same as the condition of zero wait time.

I find that the bigger wait time, more variable the trees.

However, the proposed algorithm creates the tree with ($SOL(T)$=62, $h(T)$=3, $n(P)$=5), in condition of $W = R/ D$ (s).

Now, I try to compare the performance of TPSN synchronization operations for two trees, as shown in Fig. 6, which are created by TPSN tree construction with zero wait time, and the proposed tree construction algorithm.

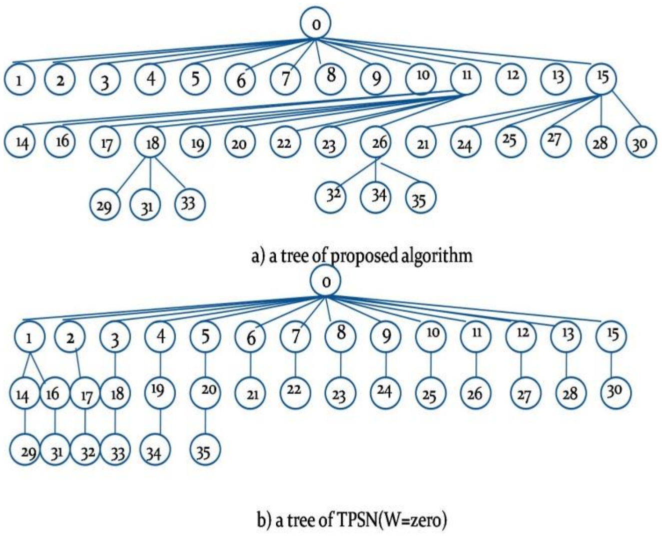

Fig. 6. Two trees created by the original and the proposed algorithms.

The power consumptions for TPSN synchronization process for a round in the two types of trees are listed in Table III. The parameters for energy model at a node in simulation are as follows; Transmission Power Consumption (0.660W), Receive Power Consumption (0.395W), Idle Power Consumption (0.035W), Initial Energy (1 Joule). And 20bytes per message was transmitted.

The result means there is 1.4% improvement in terms of energy efficiency at the proposed tree over the TPSN tree.

It is considered that power efficiency of TPSN using the tree created by the proposed algorithm is better than any tree listed in Table II.

TABLE III: TPSN PERFORMANCE COMPARISON FOR TWO TREES CREATED BY THE ORIGINAL AND THE PROPOSED ALGORITHMS.

	# of messages	Power Consumption	Remarks
Tree_o	91	15.767(J)	by original algorithm with zero wait time
Tree_p	75	15.545(J)	By the proposed algorithm

VI. CONCLUSION

TPSN, the prominent WSN time synchronization protocol, can improve the performance by using an efficient tree. An evaluation result shows that the proposed algorithm can generate better tree than TPSN's algorithm. It is expected that the proposed algorithm can be used for other time synchronization protocols which used a tree-based hierarchy.

REFERENCES

[1] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-synch protocol for sensor networks," in *Proc. ACM Sensys (the 1st Int. Conference on Embedded Networked Sensor Systems)*, 2003, pp. 138-149.

[2] D. Mill, *Computer Network Time Synchronization: the Network Time Protocol on Earth and in Space*, CRC Press, 2011.

[3] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proc. ACM OSDI the 5th Symposium on Operating Systems Design and Implementation*, 2002, pp. 1-172.

[4] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proc. ACM SenSys'04*, 2004, pp. 39-49.

[5] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: A survey," *Ad Hoc Networks*, pp. 281-323, 2005.

[6] S. Bae, "Power consumption analysis of prominent time synchronization protocols for wireless sensor networks," *Journal of Information and Processing Systems*, vol. 10, no. 2, pp. 300-313, June 2014.

[7] A. Kulaki and K. Erciyes, "Time synchronization algorithm based on timing-synch protocol in wireless sensor networks," in *Proc. Int. Symposium on Computer and Information Science*, 2008, pp. 1-5.

[8] S. Hwang and Y. Baek, "Reliable time synchronization protocol in sensor networks considering topology changes," in *Proc. Int. Workshop Distributed Computing*, 2005, vol. 3741, pp. 105–110.

[9] D. Liu, Z. Zheng, Z. Yuan, and W. Li, "An improved TPSN algorithm for time synchronization in wireless sensor network," in *Proc. Int. Conf. on Distributed Computing Systems Workshops*, pp. 279-284, 2012.

[10] The network simulator ns-2: Documentation. (November, 2011). [Online]. Available: http://www.isi.edu/nsnam/ns/ns-documentation.html

Shi Kyu Bae received the B.S. degree from Kyungpook National University, Daegu, South Korea, in 1986, in electronics. He received the M.S. and Ph.D degrees in computer engineering from Kyungpook National University in 1993, and 1998, respectively.

He has worked for Samsung Electronics Co., and joined the Dept. of Computer Engineering at Dongyang University, Korea, in 1995. He is currently serving as a professor.

Network and Communication