Enhanced Timing-Sync Protocol for Sensor Networks

Shi Kyu Bae

***Abstract*—The prominent time synchronization protocol for wireless sensor networks (WSN), *Timing-sync Protocol for Sensor Networks (TPSN)*, was developed to provide high synchronization accuracy among sensor nodes. Until now, TPSN's approach has been adopted in many other WSN synchronization schemes. However, TPSN has some room for improvement. First, performance of TPSN depends on the efficiency of the hierarchical structure, which requires more efficient structure construction. Secondly, *level discovery* process is done only once in the original TPSN. So, if one of the nodes that were deployed in the middle of the network dies due to power exhaustion, network-wide synchronization will fail.**

In this paper, a new tree construction algorithm with low complexity is proposed so that it gives better efficiency to TPSN's synchronization process. Besides, tree constructions are performed many times fully or partially, depending on network states. The proposed method's performance has been evaluated by simulation. The results show that the proposed scheme is better than one used in original TPSN.

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

I. Introduction

Wireless Sensor Networks (WSN) utilizing wireless implementation of sensor nodes have been applied to many areas such as environmental monitoring, target tracking, military surveillance, and so on. These kinds of WSN applications consider local clocks synchronized to a common reference time at each node. 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. Moreover, level discovery process is done only one time at the setup of TPSN, which means new hierarchy construction is impossible even though some problems are occurred in the network, for example, fault nodes due to power exhaustion.

Manuscript received September 26, 2014; revised March 5, 2015.

Shi Kyu Bae is with the Department of Computer Engineering, Dong Yang University, South Korea (e-mail: skbae@dyu.ac.kr).

In this paper, a new method is proposed in regards to tree construction for TPSN.

First, a new tree construction algorithm is proposed, which makes TPSN's synchronization process efficient. In addition, tree constructions are performed many times fully or partially depending on network states. It is different from TPSN, which performs level discovery only once before starting synchronization process.

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

II. Related Works

A. Existing Time-Synchronization Schemes for WSN

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.

DOI: 10.7763/IJFCC.2015.V4.380

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.

B. 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.

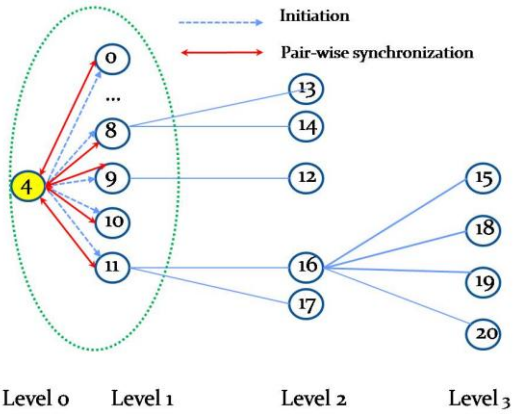

Fig. 1. A typical tree used in TPSN.

TABLE I: DEFINITION OF NOTATIONS USED IN THIS PAPER

notation	meaning
T_i	a Tree
n_i	a 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 parent nodes at a tree
$n(P)$	The number of elements in 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.

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

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

$n(P)$ ranges from 1 to ($n(T)$-1).

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. 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. The 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).

III. PROPOSED SCHEME

A. Limitation of TPSN

Although TPSN is very useful and popular time synchronization protocol, it has some limitations to be improved.

Firstly, 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.

Thus, as number of level of a node in a tree increases, clock offset of a node from the root node, which has global reference time, also increases in TPSN [2]. 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.

Secondly, level discovery process is done only once in TPSN. So, if one of the nodes that were deployed in the middle of the network dies due to power exhaustion, network-wide synchronization will fail.

B. Tree Construction Algorithm

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 *zero-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. 2, 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. 2(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).

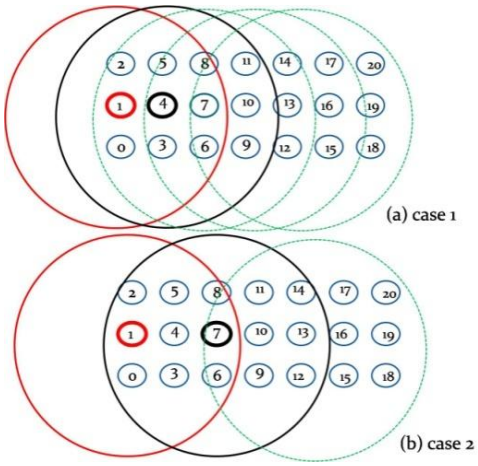

Fig. 2. Level discovery progress, (a) TPSN's case with zero-wait-time, (b) other method.

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, a new tree construction algorithm for this purpose is proposed.

In this 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 *D* is the distance between two nodes and *k* is constant, and *R* was defined in Table I.

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

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.

Second, 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.

C. Tree Reconstruction Process

The original TPSN's level discovery is done once before synchronization phase. However, the proposed method performs level discovery whenever some critical nodes request for tree reconstruction. In this case, some nodes, which do not have enough energy, do not participate in parent node determination during level discovery.

Thus, enhanced TPSN operates, other than TPSN, as shown in Fig. 3. Tree reconstruction is composed of two operations; full or partial tree reconstructions.

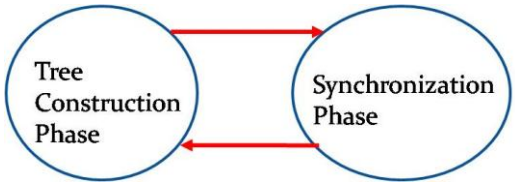

Fig. 3. Enhanced TPSN's operation.

Energy state of each node, as well as distance between nodes, is considered, whenever a full tree is reconstructed, as shown in Fig. 4.

The concept of partial tree reconstruction was proposed as Cluster Head Change procedure in [10]. When any Parent node(PN) in the tree detects that its residual energy is less than a threshold value, it tries to find another PN which is able to replace itself among normal nodes(NN) in the cluster. The candidate PNs should have the following conditions;

1) They have more residual energy than the current PN in order to act as PN for longer time.
2) They should reside in the limited and specified area, where is cross-section between the communication area of both the parent and child nodes of the current PN (shown in Fig. 5). It is because that whole tree structure of clusters should not be affected after PN is replaced with another node.

For partial tree reconstruction by replacing PN in the cluster, at least six messages are required; 'Hello', 2 times of

'Agree', several 'Request's, 'Nominate', and 'Inaugurate'.

Fig. 6 shows a procedure of the partial tree reconstruction algorithm, which consists of *step-a* through *step-d*.

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

Fig. 4. Pseudo code of the full tree construction algorithm.

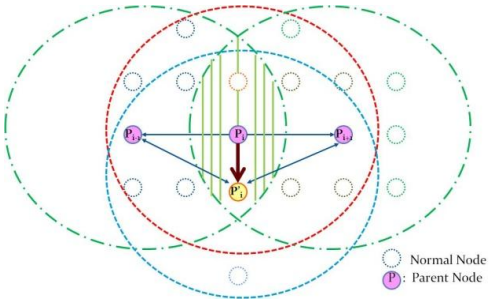

Fig. 5. Parent node change and candidates in the partial tree reconstruction.

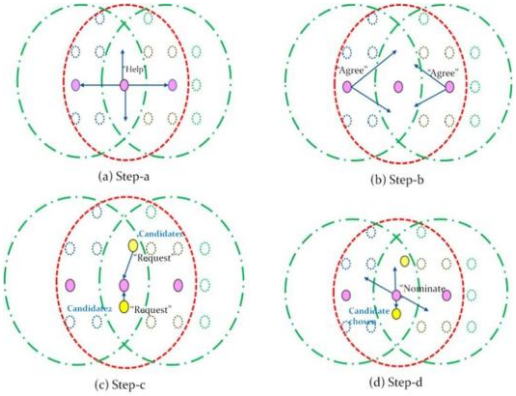

Fig. 6. Parent node change procedure [10].

1) step-a: The current PN start to find a new PN by broadcasting "Help" message with IDs of itself, its parent and child nodes.
2) step-b: Receiving "Help" message from the current PN, its parent and child nodes send broadcast "Agree" message. "Agree" messages are used for NNs to determine if they are new PN candidate[s] or not. That is to say, NNs which can receive all the messages from the current PN, its parent and its child nodes will be a candidate of a new PN.
3) step-c: Candidate[s] of a new PN check if its residual energy is more than the current PN's one. Being enough to be a new PN, they ask for a new PN by sending "Request" message to the current PN.
4) step-d: On receiving requests from the candidates for a certain interval, the current PN determines a new PN by choosing the requesting node which has the most energy value. And the current PN broadcasts the notification that the new PN is selected with "Nominate" message. After that, new PN notifies that it will begin its term after a specified time with "Inaugurate" message.

A pseudo code of the partial tree reconstruction algorithm is shown in Fig. 7.

```
<Partial Tree Reconstruction Algorithm>
Input : "HELP" packet from a Parent node(PN), p(ni)
Output : Parent node change and rejoin
.....
 If( parent or child node of current p(ni))
    send "AGREE" packet
else // normal nodes
    wait "AGREE" packets from both
  If(receive both "AGREE"s)  // new PN candidate
    participate at election or not
      after considering residual energy.
    If(selected as new PN by current PN)
       notify new PN chosen (send "INAUGURATE ")
    else // not a new PN
       Determine its PN and Join
         Switch(condition) {
           case 1 : No change ; break;
           case 2 : Change to new elected PN ; break;
           case 3 : Change to the parent of
                        the previous PN; break;
           case 4 : Change to the child
                        of the previous PN; break;
           case 5 : Find new PN and Join; break;
         }
```

Fig. 7. Pseudo code of the partial tree reconstruction algorithm.

During the parent node change, some nodes may change their roles between a normal node and a parent node. Refer to the state transition diagram from PN to NN or vice versa, as shown in Fig. 8.

PN can change its role to NN through *Help* and *Nominate* states, when it detect lack of energy. Before that, one of the NNs should be inaugurated as a new PN through *Candidate* and *Nominated* states.

If the current PN fails to change, although it tries to find a new candidate, the current PN will ask for full tree reconstruction to the root node, which was described in Fig. 4.

IV. Simulation Test

The proposed tree construction algorithm was tested by simulation with Network Simulator, NS2- 2.34 [11].

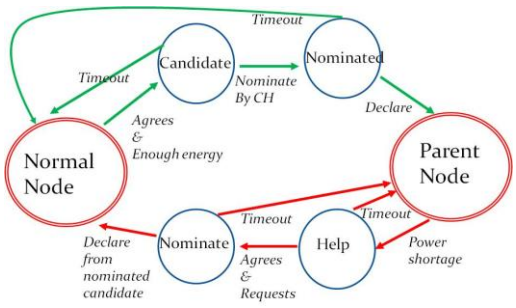

Fig. 8. State transition diagram for PN in partial tree reconstruction.

The simulation has been performed with 36 nodes in the topology shown in Fig. 9. 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 dotted 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.

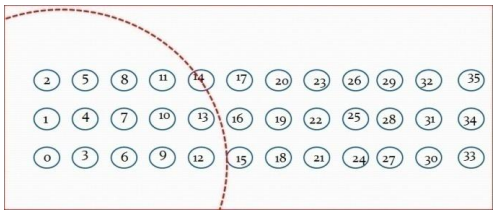

Fig. 9. A topology used in simulation.

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.

In case of zero-wait-time, the following tree was created as shown in Fig. 10(b);

$$SOL(T)=62,\ h(T)=3,\ n(P)=21.$$

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

It is considered that the bigger wait time, more variable the trees.

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

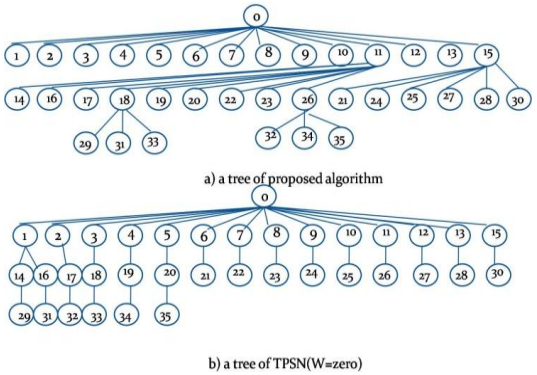

Fig. 10. Two trees created by the original and the proposed algorithms.

However, the proposed algorithm creates the tree with ($SOL(T)$=62, $h(T)$=3, $n(P)$=5), in condition of $W = R/D$ (s), as shown in Fig. 10a).

And the performance of TPSN synchronization operations for two trees was compared, which are created by TPSN tree construction with zero-wait-time, and the proposed tree construction algorithm, as shown in Fig. 10.

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.

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

	# of msg	Power Consumption	Remarks
Tree_o	91	15.767(J)	by original algo. with zero–wait-time
Tree_p	75	15.545(J)	by proposed algo.

It is considered that power efficiency of TPSN using the tree created by the proposed algorithm is better than most of trees listed in Table IV, except tree_type(e).

TABLE IV: # OF MESSAGES FOR ONE TPSN SYNCHRONIZATION ROUND AT EACH TYPE OF TREES CREATED BY THE ORIGINAL TPSN ALGORITHM

Tree types	*a*	*b*	*c*	*d*	*e*	*F*	*g*	*h*
M	80	82	78	79	74	76	77	78

V. CONCLUSION

TPSN' performance can be enhanced by the proposed approach. In addition, network life time can be prolonged by allowing tree reconstruction, if needed. 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. 1-17.

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

[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. ISCIS the 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. IWDC Int. Workshop Distributed Computing 2005, LNCS*, 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*, 2012, pp. 279-284.

[10] S. Bae, "An EIBS algorithm for wireless sensor network with life Time prolongation," *Journal of Korean Society of Computer Information(Korean)*, vol. 19, no. 9, pp. 65-73, Sep. 2014.

[11] The Network Simulator ns-2: Documentation. [Online]. Available: http://www.isi.edu/nsnam/ns/ns-documentation.html

Shi Kyu Bae received the B.S. degree in electronics from Kyungpook National University, Daegu, South Korea, in 1986. He received his 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 Department of Computer engineering at Dongyang University, Korea, in 1995. He is currently serving as a professor.