IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 2, 2017, accepted November 30, 2017, date of publication December 7, 2017,
date of current version February 14, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2780923

Density Table-Based Synchronization for Multi-Hop Wireless Sensor Networks

MAHMOUD ELSHARIEF[1,2], MOHAMED A. ABD EL-GAWAD[1,3], AND HYUNGWON KIM[1], (Member, IEEE)

[1]Department of Electronics Engineering, Chungbuk National University, Cheongju 28644, South Korea
[2]Department of Electrical Engineering, Al-Azhar University, Cairo 11651, Egypt
[3]National Telecommunication Institute, Cairo 11768, Egypt

Corresponding author: HyungWon Kim (hwkim@cbnu.ac.kr).

This work was supported by IITP grant through the Korean Government, Development of wide area driving environment awareness and cooperative driving technology which are based on V2X wireless communication under Grant R7117-19-0164.

ABSTRACT Time synchronization is becoming a general requirement for wireless sensor networks to wake up sensor nodes and achieve successful communications among the nodes. It is, however, challenging for multi-hop sensor networks to synchronize all the distributed nodes autonomously with minimum power consumption. This paper proposes a low power time synchronization protocol called density table-based synchronization (DTSync). It selects a minimal sequence of synchronization steps by estimating the density of the neighboring nodes and mapping these nodes in a density table. The proposed protocol attempts to minimize the power consumption by reducing the number of messages exchanged among the nodes. We evaluate the proposed protocol using software simulation and then real wireless networks with sensor hardware. Compared with a prior method, such as hierarchy reference broadcast synchronization (HRTS), DTSync achieves substantially fewer synchronization messages and thus lower power consumption. In addition, DTSync is guaranteed to synchronize all nodes in any multi-hop networks, whereas the previous methods are not guaranteed to synchronize all nodes. Simulation experiments demonstrate that DTSync reduces the number of synchronization messages by more than 40 times compared with HRTS for a large network of 1500 node.

INDEX TERMS Density, DTSync, HRTS, time synchronization, WSNs.

I. INTRODUCTION

To meet the cost constraints, wireless sensor nodes are often equipped with a low-cost hardware clock oscillator. Low-cost crystal oscillators tend to cause a wide range of clock drift among the nodes. Consequently, the synchronization of the network cannot be maintained for a long time, unless an accurate synchronization process is periodically conducted. In time division multiple access (TDMA) networks, loss of synchronization in any node can lead to an inaccurate wake-up time of the node, which causes serious failure of network connectivity. Therefore, sensor nodes need to exchange their time information periodically to minimize their synchronization errors caused by their clock drift [1].

Many synchronization protocols have been proposed for WSNs such as Timing-Sync Protocol for Wireless Sensor Network (TPSN) [2], Reference Broadcast Synchronization (RBS) [3], Flooding Time synchronization protocol (FTSP) [4], Hierarchy Reference Broadcast Synchronization (HRTS) and Individual-based Time Synchronization Request Protocol (ITR) [5]. The objective of these protocols is to synchronize all sensor nodes in large networks using multi-hop synchronization methods. Most of these protocols are focused on improving the synchronization accuracy but are not concerned with power [12].

In addition, WSNs are facing many challenges of maximizing the battery lifetime and reducing the data delivery latency for time critical applications [11], [16] This paper introduces a low power time synchronization protocol called Density Table based Synchronization (DTSync). It achieves low power consumption by minimizing the number of synchronization packets while ensuring that all nodes in the network are synchronized. DTSync utilizes the notion of neighbor node density to elect a set of the best reference nodes. Our goal is to cover all nodes with a minimal set of reference nodes, so it can minimize the number of synchronization messages exchanged between each reference node and its covered nodes throughout the network.

The remainder of the paper is organized as follows. In Section II, related work is discussed, while the DTSync algorithm and its procedures are elaborated in Section III.

In Section IV, Simulation is presented. The experimental results are provided in Section V followed by the discussion and future work in Section VI, and finally the conclusions in Section VII.

II. RELATED WORK

Clock synchronization has been extensively studied in the past decades. While nodes equipped with Global Positioning System (GPS) can be easily synchronized in outdoor environments [5], [12], installing GPS in every node may not always be a practical solution due to the size overhead, cost, power consumption, and indoor environment with no GPS reception. There have been many studies in the literature that are aimed at minimizing synchronization errors in WSNs.

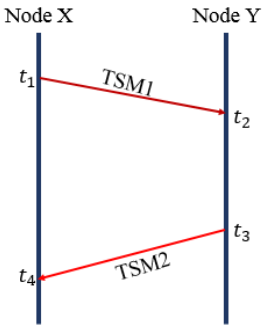

FIGURE 1. Pairwise synchronization process.

The Timing-sync Protocol for Sensor Networks TPSN [2] uses a hierarchical structure that consists of two phases. The first is a level discovery phase, which builds a hierarchical structure of the network starting from the root node. The second is a synchronization phase, where a pairwise synchronization is performed throughout the network. The pairwise synchronization process is illustrated in Fig. 1, where nodes X and Y synchronize with each other. Node X sends a Time Stamp Message1 (TSM1) to node Y at time t_1. Node Y receives TSM1 at time t_2, and then sends Time Stamp Message2 (TSM2) at time t_3 to node X. Node X receives TSM2 at time t_4, and then it calculates offset O using Eq. 1. Finally, it corrects its clock based on the offset value. Although TSPN provides high scalability, it incurs a large number of message exchanges leading to high power consumption.

$$O = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \tag{1}$$

The Reference Broadcast Synchronization (RBS) [3] aims to provide synchronization amongst a set of client nodes within the single-hop broadcast range. It attempts to remove non-deterministic portion (sent time, access time) of the critical path to ensure accurate synchronization. In RBS, however, a large number of messages are transmitted in each cycle of the synchronization. This results in two different problems, excessive energy consumption and high chance of collision [10], [13], [14]. Flooding Time Synchronization Protocol (FTSP) [4] aims to get a network-wide synchronization. It uses the advantages of both TPSN and RBS [16]. Although it incurs lower overhead than RBS, it still requires a large number of messages and thus in a dense network, it can lead to very high chance of collision. Lenzen *et al.* [8], [9] have reported that the global error of FTSP increases exponentially with the size of the network. They then proposed a scalable protocol called PulseSync which can rapidly flood the timing information throughout the network [12]. Yildirim *et al.* [1], [15] pointed out possible drawbacks of rapid flooding and proposed a clock speed agreement algorithm. It achieves high accuracy and scalability of synchronization by introducing a slow flooding method [12].

The Pairwise Broadcast Synchronization (PBS) [28] uses the pairwise operation introduced in [29]. It employs an overhearing technique to reduce the energy consumption by reducing the number of transmission messages. Multi-hop PBS has been proposed in [30], which consists of hierarchy forming and pair selection methods. Multi-hop PBS, however, needs an extra pairwise operation that degrades the energy efficiency compared with its single-hop counterpart [31]. Selecting pairs of nodes in a large network can be very costly in terms of both computation time and energy [16].

Unlike other methods, Average Time Synchronization (ATS) [20] does not require the reference node selection nor tree construction steps. Instead, ATS uses a cascade of two consensus algorithms to tune compensation parameters and allows nodes to converge to a steady state virtual clock. ATS, however, suffers from frequent collisions that causes a significant disturbance in synchronization process [16], [19]. Whereas most of these previous methods target timing accuracy and fast synchronization, most of them suffer from message loss due to frequent collisions. To our best knowledge, none of the previous methods attempted to preventing collisions entirely.

Rather than executing the pairwise synchronization process among all nodes, HRTS [5] has a simpler process. In what follows, we describe the operation of HRTS, since it provides the basis for our proposed method's time offset calculation.

It consists of three simple steps that are repeated in each hierarchical level over a multi-hop network. In HRTS, it is assumed that each node knows its surrounding neighbors. Fig. 2 illustrates the operation of HRTS. In the first step, a Reference Node (RN) broadcasts a synchronization message with time stamp t_1 as in Fig. 2(a). The neighboring nodes record their time stamp t_2 when they receive the message from RN as shown in Fig. 2(a). One child node, e.g. N2, is randomly pre-selected by RN to send a reply message carrying t_{2r} (subscript r is added to distinguish the selected node from the other nodes) and t_3 (the time right before transmitting the reply message) as shown in Fig. 2(b). Once RN collects all time stamps t_1, t_{2r}, t_3 and t_4 (the time right after receiving the reply message), RN calculates the clock

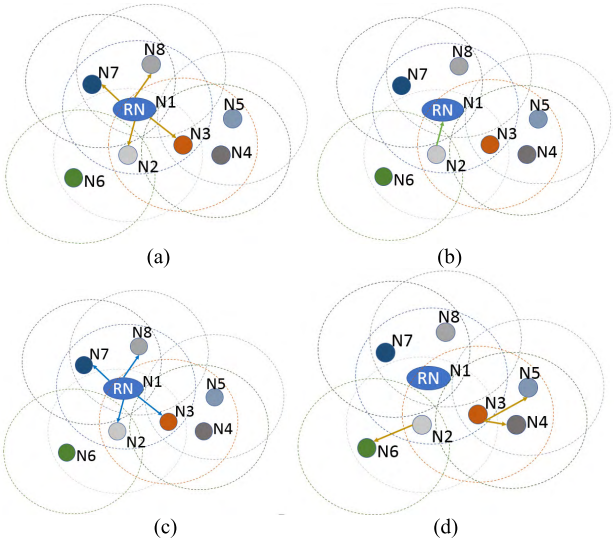

FIGURE 2. Multi-Hop Synchronization for HRTS: (a) Reference Node (RN) N1 broadcasts a synchronization message. (b) N2 Transmits a replay message with t2r and t3 (c) RN broadcasts again to all neighbor nodes with Offset time and t2r. (d) N2 and N3 repeats the above procedure for synchronization of the next hop.

offset O using Eq. 2.

$$O = \frac{(t_{2r} - t_1) - (t_4 - t_3)}{2} \tag{2}$$

RN then broadcasts t_{2r} and O to its child nodes as shown in Fig. 2(c). Child nodes, N2, N3, N7 and N8, compare their arrival time stamp t_2 with the received arrival time stamp t_{2r}. For example, N3 calculates the offset difference d by Eq. 3.

$$d = t_{2r} - t_2 \tag{3}$$

Finally, the clock of N3 corrects its old clock by using Eq. 4.

$$t^{new} = t^{old} + O + d \tag{4}$$

This process is repeated in the subsequent hierarchy levels downwards from the base station node RN as shown in Fig. 2(d) [5]. The overhead (expressed by the number of packet exchanges) of HRTS and RBS are compared by Dai *et al.* [5]. While HRTS imposes much lower overhead than RBS, the overhead of HRTS increases when the node density increases. Also, HRTS relies on the classical CSMA protocol to transfer its messages, and so it can incur excessive collisions. We can conclude that HRTS has two drawbacks: higher overhead and excessive collisions.

In [7], the concept of scheduling based on density was introduced, which can reduce the above overhead by decreasing the number of hierarchy levels. It can, however, still suffer from the problem of frequent collisions and reception failure, since the child nodes can send reply messages to their reference node at the same time. In addition, it often has a coverage problem, since it does not guarantee synchronization of certain nodes that are not in the wireless range of any reference nodes elected. It, therefore, requires a second-pass scheduling process to cover the unsynchronized nodes.

The proposed DTSync algorithm also employs a scheduling process based on neighbor density to achieve low overhead. It introduces a new technique that avoids collisions entirely and guarantees that 100% of the nodes are synchronized in a single-pass scheduling process.

III. DTSync ALGORITHM

DTSync is a multi-hop synchronization algorithm aimed to reduce the power consumption while synchronizing the entire WSN. It provides an accurate and low power approach to the goal of guaranteed synchronization for mesh or tree-based network topologies. DTSync consists of three processes: (1) a neighbor discovery process, (2) a scheduling process, and (3) a periodic synchronization process. The neighbor discovery and scheduling processes run only once during the setup phase or when the network topology has a significant change. The synchronization process is conducted periodically with a time period of P_{syn} to maintain the time offset of all nodes within an acceptable range. The proposed method can take the initial sensor network in any general form of mesh network. Once the scheduling process is complete, the resulting network topology becomes a tree structure with the sink node (Pre-defined) acting as the root. In the tree, all the elected reference nodes form a stem of the tree, while all other nodes form branches of the tree. During the periodic synchronization process, this tree structure does not change [18], [23]. The following sections describe each of the processes in detail.

A. NEIGHBOR DISCOVERY PROCESS

When the nodes of a WSN startup, they broadcast a Neighbor Discovery Message (NDM) to discover their neighbor nodes in one-hop range. As the underlying MAC protocol of this process is a conventional CSMA protocol, collisions can occur when multiple nodes broadcast their NDM messages at the same time. In the proposed neighbor discovery process, each node sends 3 NDMs at random times during discovery time t_{disc} (e.g. 30 seconds) to ensure that each neighbor node receives at least one NDM. Upon receiving an NDM, each receiver sends an ACK message back to the sender. After this discovery process, each sender node can determine all its neighbors located within its wireless range. A sender node then assigns to each neighbor a Wait Time Quantity (WTQ) in the order by which it receives the ACK messages from the neighbor nodes. Finally, the sender node notifies each of its neighbor nodes of the assigned WTQ. WTQ is a slotted delay time for which each node waits before transmitting a reply message to avoid collisions. WTQ is similar to the TDMA time slots, and it is defined by Eq. 5.

$$WTQ > D_{pg} + D_{Tx} + D_{pc} + G_I \tag{5}$$

Here D_{pg}, D_{Tx}, and D_{pc} are propagation, transmission and processing delays, respectively. G_I is a guard interval that indicates allowed time discrepancy between a transmitter and a receiver node [21]. In the experiment with the sensor hardware, the packet length was 120 Bytes (960 bits), and the

date rate was 50Kbps, which gives a transmission time $D_{Tx} = 19.2ms$. We chose $D_{pg} + D_{pc} = 0.4ms$, and $G_I = 0.4ms$. The above parameters used in our experiment determined the minimum WTQ as 20ms [26], [27].

The proposed method assigns a unique WTQ to each neighbor node, so each node is guaranteed to transmit the reply message to the sender node at a different time. If any node received NDM but did not receive a WTQ, the node concludes that its prior ACK Message has been lost due to a collision, and so it sends ACK Message again. Table 1 shows an example WTQ table constructed for the example network of Fig. 3. In Table 1, each node assigns unique WTQs to all its neighbor nodes. Suppose that N1 received ACK messages from its neighbors in the order: N2, N3, N7, N8. Hence N1 assigns WTQs of 1 to N2, 2 to N3, 3 to N7, and 4 to N8. As a result of this process, each node builds two tables. One is a pseudo Time Division Multiplexing (TDM) table, while the other is a density table.

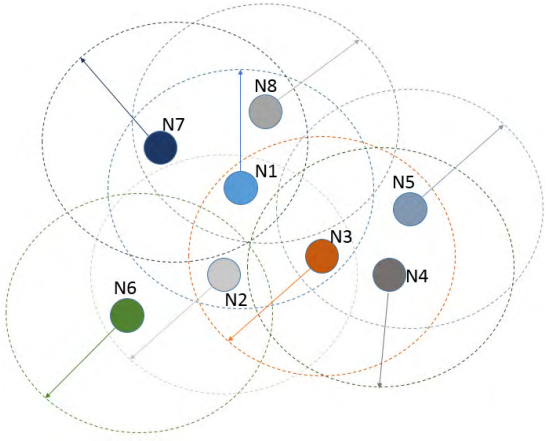

FIGURE 3. Wireless Sensor Network example.

TABLE 1. WTQ Table constructed for all nodes in Fig. 3 by assigning time slots.

Sender / Neighbor (Receiver)	N1	N2	N3	N4	N5	N6	N7	N8
N1		1	1				1	1
N2	1		2			1		
N3	2	2		1	1			
N4			3		2			
N5			4	2				
N6		3						
N7	3							2
N8	4						2	

1) PSEUDO TIME DIVISION MULTIPLEXING (TDM)

A pseudo TDM table contains a list of neighbor nodes with Wait Time Quantity (WTQ). Once complete pseudo TDM tables are obtained for all nodes, the scheduling process can entirely avoid any collisions by using the pseudo TDM tables.

TABLE 2. Pseudo TDM tables for all nodes in Fig. 3

Sender / Receiver	N1	N2	N3	N4	N5	N6	N7	N8
N1		1	2				3	4
N2	1		2			3		
N3	1	2		3	4			
N4			1		2			
N5			1	2				
N6		1						
N7	1							2
N8	1						2	

Table 2 shows example pseudo TDM tables constructed from the neighbor tables of Table 1. The rows of Table 2 correspond to the sender nodes, while the columns indicate the receiver nodes. Each row contains the WTQ values that have been assigned by the sender node to its neighbor nodes as in Table 1. It can be observed that Table 2 is a transposed form of Table 1. For example, the column for node N1 (highlighted) in Table 1 is transposed to form a row for sender N1 (highlighted) in Table 2. Each column corresponds to the pseudo TDM table constructed and maintained by each receiver node.

In fact, each node keeps just one table (only one corresponding column in Table 2) for its neighbors only, and determines WTQ wait time for sending reply messages to its neighbors using its pseudo TDM table. Hence the memory overhead for the tables is negligible, since each table contains only the one-hop neighbor information, and furthermore this memory is removed after the scheduling process which is conducted only once in the lifetime of the network. For example, the numbers 2, 2, 1, 1, in the column of receiver N3 of Table 2 indicate that N3 waits WTQs of 2 units before N3 transmits the reply messages to N1 and N2, while waiting WTQ of 1 unit for the replies to N4 and N5, respectively. The empty entry in the column means that there is no neighbor relationship between the two nodes. Using the pseudo TDM tables, the scheduling process described below conducts a pseudo TDM protocol by allocating a unique transmit time slot to all acknowledgement messages. This way, the pseudo TDM protocol can ensure that no collision occurs.

2) DENSITY TABLE

To build a density table in each node, the node makes a copy from pseudo TDM table and then replaces WTQ by 1 in the copied table. Table 3 shows a density table constructed in this way from the pseudo TDM table of Table 2. The columns of Table 3 correspond to the density table for each node. The empty entries under column N_i mean that there is no neighbor relationship between the two nodes, while the entries of value 1 under N_i are added together to calculate the current density metric of node N_i. When N's neighbor node, say N_j, has been covered by any reference node, N_j's corresponding entry is set to 0. Such the entries of value 0,

TABLE 3. Density tables for all nodes in Fig. 3 constructed by neighbor discovery process.

Neighbor \ Nodes	N1	N2	N3	N4	N5	N6	N7	N8
N1		1	1				1	1
N2	1		1			1		
N3	1	1		1	1			
N4			1		1			
N5			1	1				
N6		1						
N7	1							1
N8	1						1	

therefore, would not contribute to the density metric of N_i in the next scheduling process. Density tables constructed in this way are used for scheduling process described below.

3) ANALYSIS OF NEIGHBOR DISCOVERY PROCESS

In this section, we analyze the complexity of neighbor discovery process. There are three types of messages NDM, ACK, and WTQ. Suppose that the total number of nodes is N, while node i is a sender node and $\mathbb{V}(i)$ indicates a set of neighbor nodes of node i. Then $\mathbb{V}(i)$ can be expressed by Eq. 6.

$$\mathbb{V}(i) = \left\{v_{i,1}, v_{i,2}, .., .., v_{i,j}, .., .., v_{i,N}\right\}, \quad \forall i = 1 : N \quad (6)$$

Here, $v_{i,j}$ is defined by

$$v_{i,j} = \begin{cases} 1, & \textit{if node j is in the wireless range of node i} \\ 0, & \textit{otherwise} \end{cases}$$

The total number of NDMs is given by Eq. 7.

$$\textit{The number of NDMs} = \alpha N \quad (7)$$

Here α is the number of NDMs sent by node i. The total number of ACKs is given by Eq. 8.

$$\textit{The number of ACKs} = \sum_{i=1}^{N} \sum_{j=1}^{N} \beta_{i,j} v_{i,j} \quad (8)$$

Here β is the number of *ACKs* sent by node i. The total number of WTQ messages is given by Eq. 9.

$$\textit{The number of WTQs} = \sum_{i=1}^{N} \sum_{j=1}^{N} \gamma_{i,j} v_{i,j} \quad (9)$$

Here γ is the number of *WTQs* sent by node i. The total number of messages transmitted during the neighbor discovery process, therefore, is given by Eq. 10.

$$\textit{The number of all messages} = \alpha N + \sum_{i=1}^{N} \sum_{j=1}^{N} (\beta_{i,j} v_{i,j} + \gamma_{i,j} v_{i,j}) \quad (10)$$

In fact, NDM is a broadcast message, and so it does not have acknowledgment. Thus, it may be lost due to many reasons (hidden node problem, bad channel, ...) [21], [24], [25]. In our implementation, therefore, we transmit α NDMs which ensures the reliable delivery of NDMs. The empirical parameter α can be selected depending on the channel condition and the average distance between the nodes. In this paper, we chose $\alpha = 3$ based on the following experiments. We experimented a range of numbers from 2 to 7 for the retransmission limit α to determine the optimal number without incurring excessive overhead. We discovered that the number greater than 3 imposed reasonably small overhead, while the number 3 gave us equally good performance. In our simulation, we figured out the number of all required messages. For example, the large network of 1000 nodes (area: 1000 meters $\times$ 1000 meters, wireless range radius: 25m) had the maximum number of messages calculated by Eq. 10 is 6796 messages. For such a large network, the number of messages needed is about 7 messages per node. This is a small overhead, since messages are transmitted only once during the neighbor discovery process.

B. SCHEDULING PROCESS

The scheduling process is an iterative algorithm for selecting an ordered set of reference nodes. As in HRTS, the reference nodes are the master nodes that exchange time synchronization messages with their neighbor nodes and spread their timing information throughout the entire network [5]. The key idea behind the scheduling process is to find a minimal sequence of reference nodes that can cover all the nodes in the network. This ensures that the periodic synchronization process can synchronize the entire network with the minimal number of message exchanges, and consequently minimal energy consumption. The scheduling process consists of two sub-processes, 1) Forward-Trace Scheduling and 2) Backward-Trace Scheduling Processes as described below.

1) FORWARD-TRACE SCHEDULING PROCESS (FTS)

FTS process finds a minimal sequence $\boldsymbol{S}$ of multi-hop reference nodes starting from the sink node i such that $\boldsymbol{S}$ covers all nodes. FTS is conducted only when the network is newly installed or the network experiences significant changes like existing nodes are relocated or new nodes are added. FTS process is conducted using the pseudo-TDM protocol based on WTQs, which have been assigned in the neighbor discovery process. The transmitter in each hop sets the initial time, while the responders count their timer starting from this initial time, and they send the density information back to transmitter upon their timer indicates their WTQ time slot. By using the pseudo-TDM protocol, FTS process can avoid collisions entirely for all broadcast messages.

A flow diagram of FTS is illustrated by Fig. 4, while a detailed procedure of FTS is described below. As the first step, sink node i broadcasts a Schedule Request Message (SRM) to its neighbor nodes. Consequently, these nodes propagate the same message to the entire network by multi-hop forwarding.

The reference election algorithm starts with the sink node as the first reference node $i = Sink$. Let's assume that the network has N nodes. We denote a node k as covered, when FTS elects a reference node i, and k is within the wireless range of i, and so k can be synchronized by i.

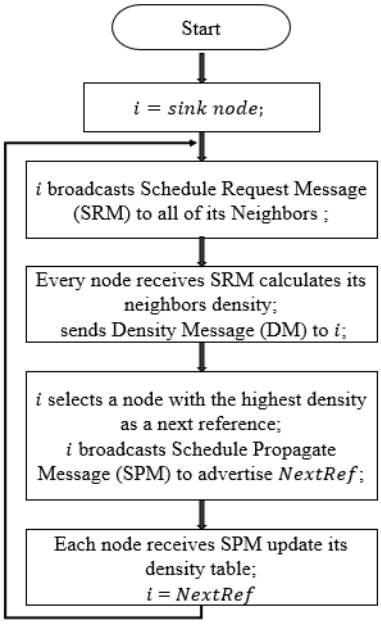

FIGURE 4. Flow chart of Forward-Trace Scheduling process (FTS) and electing the next reference node.

Step 1: The current reference node i broadcasts a Schedule Request Message (SRM) to the nodes in $\mathbb{U}(i)$. Let $\mathbb{V}(i)$ be the set of nodes that are within the wireless range of i. Let $\mathbb{U}(i)$ be the set of nodes in $\mathbb{V}(i)$ that are not covered yet. $\mathbb{V}(i)$ and $\mathbb{U}(i)$ are expressed by Eq. 6 and Eq. 11.

$$\mathbb{U}(i) = \{u_{i,1}, u_{i,2}, \ldots, u_{i,j}, \ldots.., u_{i,N}\} \quad \forall j = 1 : N \tag{11}$$

Where $\mathbb{U}(i) \subset \mathbb{V}(i)$,

$$u_{i,j} = \begin{cases} 1, & \textit{if node } j \textit{ is still not covered} \\ 0, & \textit{otherwise} \end{cases}$$

Step 2: For each node j that receives SRM, j calculates a density metric D_j by counting the number of uncovered neighbors in the density table, and then j broadcasts a Density Massage (DM) that contains D_j. Here, the density table is constructed by using Eq. 12, while the density metric D_j is calculated by Eq. 13.

$$Den(j) = \left\{d_{j,1}, d_{j,2}, \ldots.., d_{j,k}, \ldots.., d_{j,N}\right\}, \quad \forall k = 1 : N \tag{12}$$

Where

$$d_{j,k} = \begin{cases} 1, & \textit{if node } k \in \mathbb{U}(j) \\ 0, & \textit{otherwise} \end{cases}$$

$$D_j = \sum_{k=1}^{N} d_{j,k} \tag{13}$$

Step 3: Node i gathers D_j from all its neighbor nodes j's in $\mathbb{U}(i)$, and chooses the next reference node R_{i+1} using Eq. 14.

$$NextRef = MaxDensity(\mathbb{U}(i)) \tag{14}$$

Here, $MaxDensity(\mathbb{U}(i))$ selects the node whose D_j is the largest among $\mathbb{U}(i)$.

Step 4: Node i broadcasts a Schedule Propagate Message (SPM) to $\mathbb{U}(i)$. SPM advertises that *NextRef* has been elected as the next reference node, so all nodes in $\mathbb{U}(i)$ can update their density tables using Eq. 5 and 6.

Step 5: The next reference node *NextRef* sets itself as the current reference node i, and iterates from ***Step* 1** to ***Step* 5**.

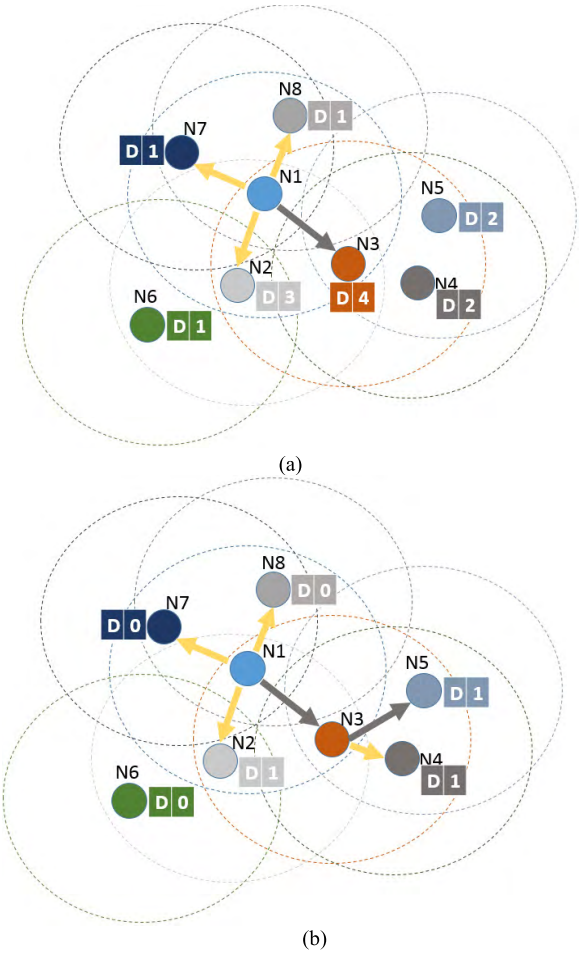

FIGURE 5. Example WSN illustrating the Forward-Trace Scheduling process (FTS): (a) FTS process for first hop. (b) FTS process for second hop.

The process is repeated until no further reference node can be selected. In each iteration, all covered nodes are removed from the density table by setting the entry to 0. By removing such nodes, it can avoid duplicate calculations of density, so the next election operations are accurate, and provide optimal scheduling results in the end. Fig. 5 shows an example of the FTS process applied to a small network with two hops. Table 4 and Table 5 show the density tables updated by FTS process, in the 1st and 2nd iterations, respectively.

2) BACKWARD TRACE SCHEDULING PROCESS (BTS)

After completing the FTS process, we have a complete sequence $R_1 \rightarrow R_2 \rightarrow \ldots. \rightarrow R_H$ of reference nodes. Some nodes, however, may be still left out without being covered by any reference node. This occurs if a node is not in the wireless range of any of the elected reference nodes. Such nodes

TABLE 4. Density tables for all nodes in Fig. 5 updated by the FTS process for the first hop level.

Neighbor \ Nodes	N1	N2	N3	N4	N5	N6	N7	N8
N1		0	0				0	0
N2	0		0			0		
N3	0	0		0	0			
N4			1		1			
N5			1	1				
N6		1						
N7	0							0
N8	0						0	

would remain unsynchronized during the periodic synchronization process. Thus, we define them as an "uncovered node". For example, N6 in Fig. 5 (b) is an uncovered node. To eliminate such uncovered nodes, we introduce the following two methods.

Method 1: In the first method, after a given time T_{exp} expires, all uncovered nodes check if they are not covered yet. If so, such nodes send a request message to their neighbor nodes. Any covered neighbor that receives this request can act as a "local reference node". While this method guarantees that all nodes get covered, it tends to incur a large number of message exchanges if more than one neighbor attempt to serve as a local reference node. Furthermore, it is difficult to determine the minimum time T_{exp}. Fig. 6 (a) shows an example of Method 1 applied to the example network of Fig. 5.

Method 2: In this method, to start the discovery of the uncovered nodes, the last reference node R_H broadcasts a Trace Back Message (TB_{MH}). TBM is relayed through all previous references in the backward direction $R_H \rightarrow R_{H-1} \rightarrow \ldots. \rightarrow R_1$. Upon receiving a TBM_k, each covered node N_i that is covered by a reference node R_k, checks its density table. If there is any remaining neighbor N_u indicated by "1" in the table, this confirms that N_u is not covered yet. Then N_i is assigned as a local reference node for N_u. N_i is responsible for providing a synchronization message to N_u. Then N_i broadcasts SRM to N_u. Once N_u receives the SRM, N_u now turns to a covered node. Then, N_u broadcasts an Acknowledge (ACK) message to announce to all its neighbor nodes that N_u has been just covered. Every node that receives the ACK message updates it density table by setting N_u's entry to 0. This way, Method 2 prevents N_u from getting more than one SRM from different local references.

The above process is repeated for all N_u's found by all N_i's assigned as local reference nodes. For example, Fig.6(b) shows that N6 is an uncovered node. N2 finds that N6 is still marked as uncovered in N2's density table, which is shown in Table 5. Hence N2 assigns itself as a local reference node and covers N6. In fact, in certain rare cases, collisions can occur in BTS Process. Suppose that in the backward-trace process, two covered nodes declared themselves as local reference nodes because they still have uncovered nodes in their

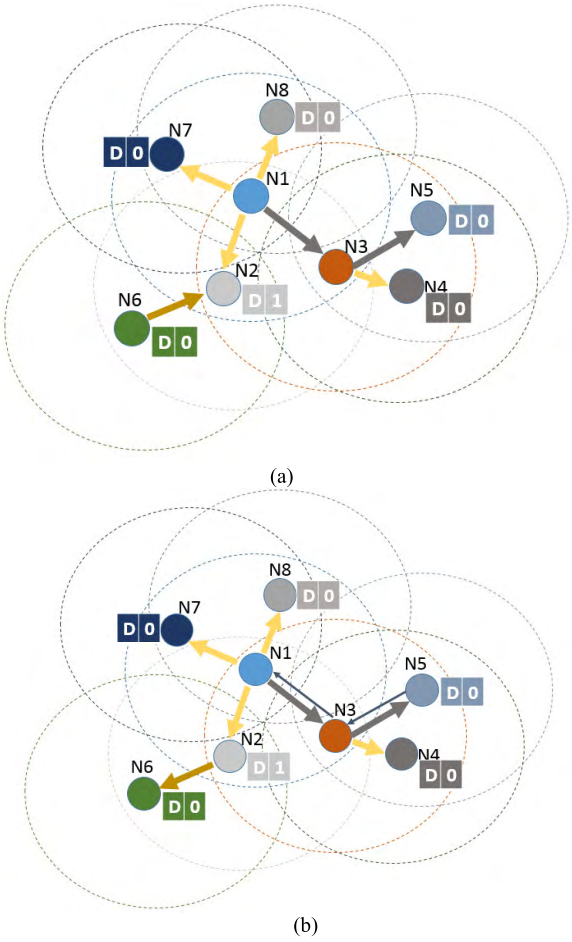

FIGURE 6. Two methods of covering uncovered node in an example WSN, when N6 was left uncovered (a) Method 1: N6 sends a request message for synchronization. (b) Method 2: N2 checks its density table after receiving a trace back message and acts as a local reference to cover N6.

TABLE 5. Density tables for all nodes in Fig. 5 updated by the FTS process for the second hop level.

Neighbor \ Nodes	N1	N2	N3	N4	N5	N6	N7	N8
N1		0	0				0	0
N2	**0**		0			0		
N3	**0**	0		0	0			
N4			0		0			
N5			0	0				
N6		1						
N7	**0**							0
N8	**0**						0	

density table. If these two local reference nodes are out of range from each other, they may attempt to send their messages to the same uncovered node at the same time. In this case, a collision may happen [21], [22]. Even if a collision occurs, however, the two nodes would back off and retransmit their messages with different random back-off delays. This is because the two nodes would still find the uncovered node in

their density table until it is covered. This way, it is guaranteed that any remaining uncovered nodes would be covered and removed from density tables eventually.

The advantages of method 2 over method 1 include: (1) it incurs fewer message exchanges, (2) it uses a deterministic time to start BTS process, and (3) TBM is used for triggering the periodic synchronization process. Due to these advantages, we chose method 2 for the implementation of DTSync protocol.

C. PERIODIC SYNCHRONIZATION PROCESS

Once the scheduling process elects a complete sequence $R_1 \rightarrow R_2 \rightarrow \ldots. \rightarrow R_H$ of reference nodes including local reference nodes, the periodic synchronization process synchronizes every node in each period of P_{sym}.

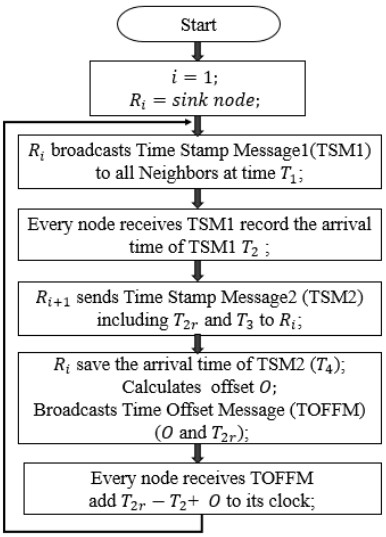

FIGURE 7. Flow chart of the periodic synchronization process.

Fig. 7 illustrates a flow diagram of the Periodic Synchronization [17]. Its detailed procedure is described as follows.

Step 1: Set $R_i = R_1$ (the first reference node).

Step 2: R_i broadcasts Time Synchronization Message 1 (TSM1) at time T_1 (R_i) to all its neighbor nodes $j \in \mathbb{U}$ (R_i).

Step 3: Each node $j \in \mathbb{U}$ (R_i) records the time stamp for the arrival time $T_2(j)$ of TSM1.

Step 4: R_{i+1} (the next reference node) sends Time Synchronization Massage 2 (TSM2) back to R_i. TSM2 contains the time stamp for the transmit time T_3 (R_{i+1}) of TSM2 as well as the arrival time T_2 (R_{i+1}) of TSM1.

Step 5: R_i records the time stamp for the arrival time $T_4(R_i)$ of TSM2. R_i then estimates the clock offset $O_{R_i,R_{i+1}}$ between R_i and R_{i+1} using Eq. 15 which corresponds to HRTS's Eq. 2.

$$O_{R_i,R_{i+1}} = \frac{(T_2(R_{i+1}) - T_1(R_i)) - (T_4(R_i) - T_3(R_{i+1}))}{2} \tag{15}$$

Step 6: R_i broadcasts to all its neighbor nodes $j \in \mathbb{U}$ (R_i) a Time Offset Message (TOFF) which contains time T_2 (R_{i+1}) and $O_{R_i,R_{i+1}}$.

Step 7: Each node j in $\mathbb{U}(R_i)$ receives TOFF, and estimates the offset difference $d_{R_{i+1},j}$ using Eq. 16 which corresponds to HRTS's Eq. 3.

$$d_{R_{i+1},j} = T_2(R_{i+1}) - T_2(j) \tag{16}$$

Then each node corrects its clock by adding the offset $(d_{R_{i+1},j} + O_{R_i,R_{i+1}})$ to its old clock using Eq. 17.

$$t_j^{new} = t_j^{old} + O_{R_i,R_{i+1}} + d_{R_{i+1},j} \tag{17}$$

Step 8: Set $R_i = R_{i+1}$. Then repeat Step 2 through Step 8 until R_i reaches the final reference node R_H.

The periodic synchronization process requires only three broadcast messages for each reference node, no matter how dense or large the network is. In addition, the scheduling process minimizes the set of elected reference nodes. The DTSync can, therefore, substantially reduce the number of message exchanges, and so it is well suited to low power applications.

IV. SIMULATIONS

This section presents simulation results to compare the performance of DTSync with HRTS. We developed a simulator that conducts the algorithms of both DTSync and HRTS.

A. EXAMPLE NETWORKS FOR SIMULATION

Simulations have been conducted in MATLAB using wireless networks of various sizes. In each network, nodes are randomly located within an area of 100m × 100m. The sink node is located at the center of the network. The following are assumed in the simulation experiments for the sake of simplicity to evaluate the proposed scheduling algorithm. Each node is equipped with an omnidirectional antenna and has a fixed wireless range of 25 meter radius. A node can communicate with all the nodes within its wireless range in line of sight, i.e., with no obstacles. It is assumed that the wireless channel is ideal, and while all the nodes are stationary.

B. EVALUATION OF SYNCHRONIZATION SCHEDULING ALGORITHM

To illustrate the benefit of using the density table while choosing a reference node, two versions of DTSync protocols are compared: a density-based method and a random-selection method. In the density-based method, as described in Section III.B, the scheduling process selects as a reference node the node whose density is the highest. In the random-selection method, on the other hand, the reference nodes are randomly selected among the neighbor nodes. The two methods otherwise employ the same procedures for the rest of the protocol. Table 6 and 7 compare the simulation results of the two methods.

Table 6 compares the results of FTS process, while Table 7 compares the results of BTS process between the density-based selection and random selection methods. In Table 6, "Reference nodes" indicate the number of nodes that have

TABLE 6. Comparison of FTS Process of DTSync Using "Density-Based Selection" and "Random Selection"

Test Network	No. of Nodes	Density Based Selection		Random Selection	
	N	Reference Nodes	Uncovered Nodes	Reference Nodes	Uncovered Nodes
Network1	100	17	0	7	49
Network2	200	16	2	9	74
Network3	500	15	11	13	104
Network4	1000	14	18	16	81
Network5	1500	14	15	16	161
Network6	2000	14	19	16	132

TABLE 7. Comparison of BTS process of DTSync Using "Density-Based Selection" and "Random Selection"

Test Network	No. of Nodes	Density Based Selection		Random Selection	
	N	Local Reference Nodes	Uncovered Nodes	Local Reference Nodes	Uncovered Nodes
Network1	100	0	0	44	0
Network2	200	2	0	61	0
Network3	500	11	0	101	0
Network4	1000	18	0	80	0
Network5	1500	15	0	160	0
Network6	2000	19	0	130	0

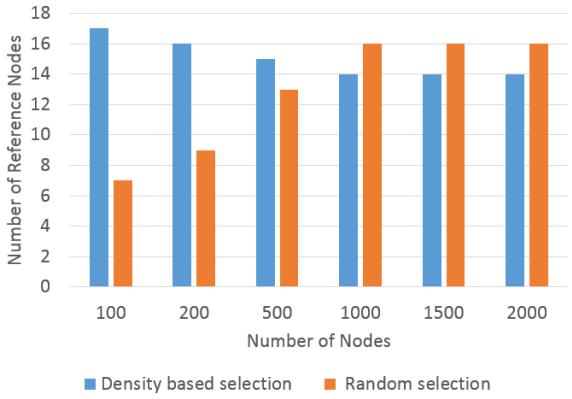

FIGURE 8. Number of reference nodes elected by DTSync FTS process using Density based selection and Random selection methods.

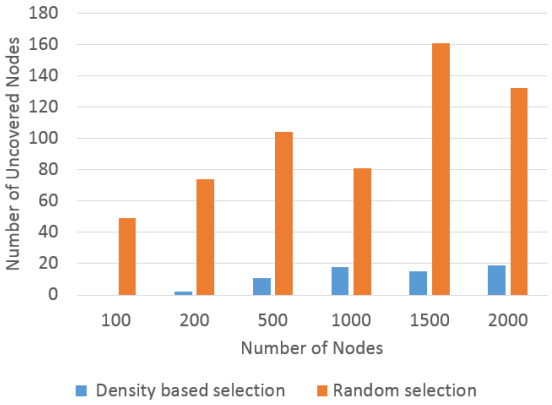

FIGURE 9. Number of uncovered nodes for DTSync FTS process using Density based selection and Random selection methods.

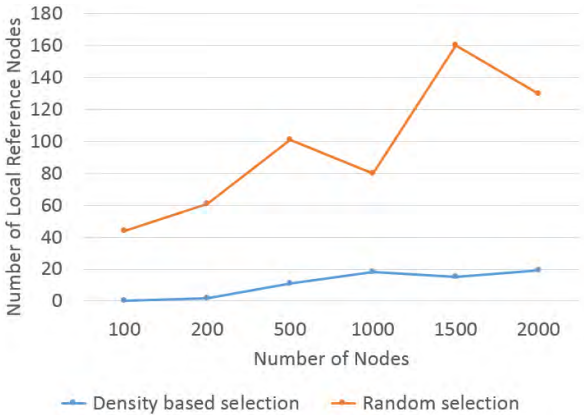

FIGURE 10. Number of Local Reference nodes elected by DTSync BTS process using Density based selection and Random selection methods.

been selected as reference nodes by FTS process. "Uncovered nodes" depict the number of nodes that are left uncovered after FTS process. "Local reference nodes" in Table 7 denote the number of nodes that have been selected as local reference nodes by BTS process to cover all uncovered nodes.

This comparison is also illustrated by Fig. 8 ∼ Fig. 10. Fig.8 compares the number of reference nodes selected by FTS process, while Fig. 9 compares the number of uncovered nodes left after FTS process. Fig. 10 compares the number of local reference nodes needed by the BTS process. These results demonstrate that DTSync with density-based selection produces significantly fewer uncovered nodes and consequently fewer local reference nodes needed. As the random-selection method tends to give different results in each simulation run, we have taken an average result from 10 simulation runs for each example network. Fig. 11 illustrates the result of DTSync for network1 (with 100 nodes). Fig.11(a) gives the result of the density-based selection, and (b) the random selection. In both (a) and (b), the red links represent the sequence of the reference nodes elected by the FTS process, while the green links denote the coverage of neighbor nodes covered by the elected reference nodes. The blue links represent the coverage of nodes covered by the local reference nodes that are selected by BTS process.

Here, note that for the density-based selection in Fig. 11(a), FTS covers all nodes leaving no uncovered nodes. In contrast, the random selection in Fig. 11(b) leaves 49 uncovered nodes, and thus it added 44 local reference nodes in the BTS process. In this simulation experiment, we evaluated the performance of the scheduling process only. The performance of the periodic synchronization process is analyzed in Section IV.C.

C. EVALUATION OF PERIODIC SYNCHRONIZATION

To evaluate the periodic synchronization process of DTSync, we have also implemented HRTS algorithm and compared the results of DTSync with HRTS. Table 8 shows the results of three methods: DTSync with density based selection,

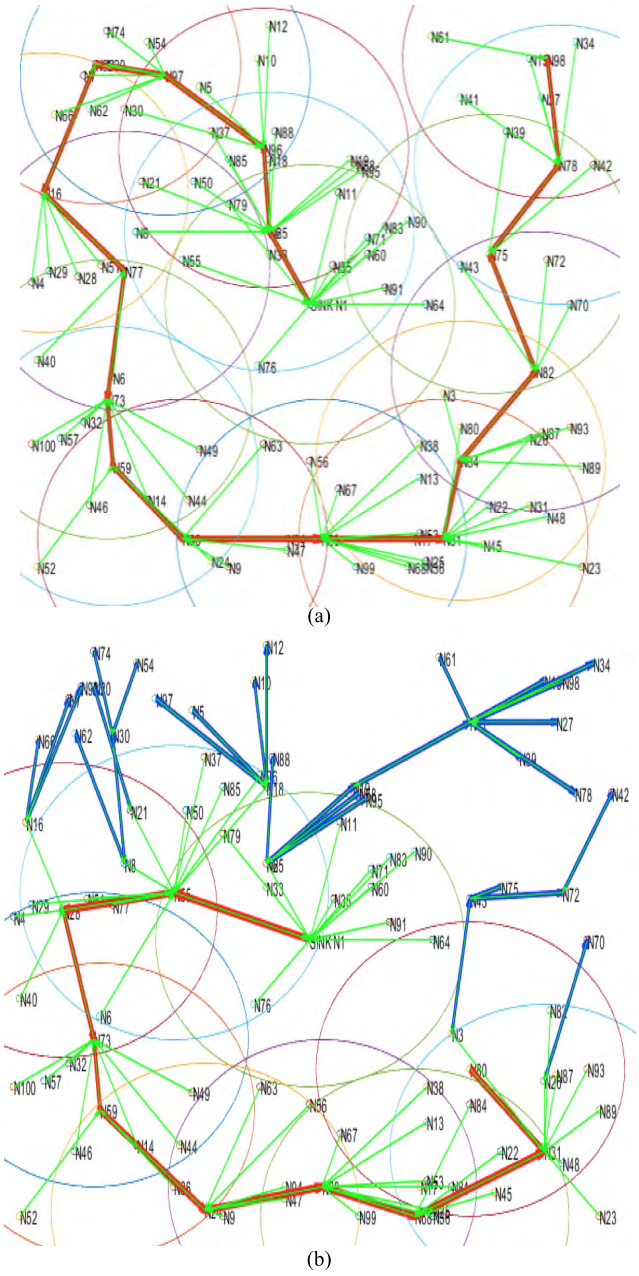

FIGURE 11. The results of DTSync scheduling process using (a) Density-based selection, and using (b) Random-selection method. For an example network, Network1 with 100 nodes is used.

DTSync with random selection, and HRTS. We measured the number of timing message exchanges during the periodic synchronization process of DTSync for the six networks used above.

As described in Section II, HRTS often incurs a large number of collisions and so it tends to leave many nodes uncovered, while DTSync's scheduling process is guaranteed to have no collisions. For the sake of simplicity, we assume that no collision happens even in HRTS. Table 8 shows that DTSync has substantially fewer message exchanges than HRTS. DTSync with density-based selection gives the better

TABLE 8. Comparison between DTSync and HRTS in the number of Timing Message exchanges.

Test Network	No. of Nodes	No. of Timing Message exchanges		
	N	DTSync with Density Based Selection	DTSync with Random Selection	HRTS
Network1	100	51	153	201
Network2	200	54	210	378
Network3	500	78	342	1104
Network4	1000	96	288	2274
Network5	1500	87	528	3543
Network6	2000	99	438	4659

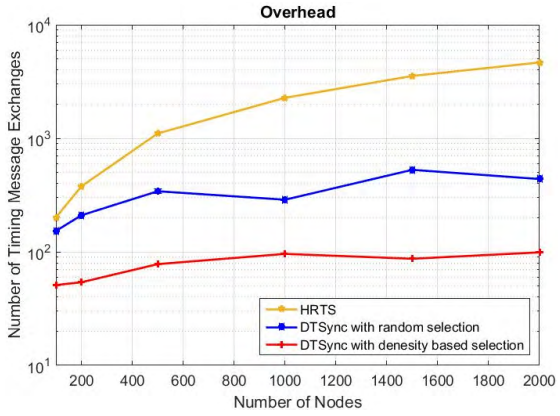

FIGURE 12. Timing message exchanges vs. number of sensor nodes.

results especially when the node density is high in the network (or large networks).

Fig. 12 compares the number of messages required by the three synchronization methods over various network sizes from 100 nodes to 2000 nodes: (1) DTSync with density based selection, (2) DTSync with random selection, and (3) HRTS. DTSync with density based selection substantially outperforms the other algorithms for large networks. For the network of 2000 nodes, DTSync with density based selection transmitted only 99 messages, while DTSync with random selection transmitted 438 messages (4 times more), and HRTS transmitted 4659 messages (47 times more than DTSync with density based selection).

Each message exchange consists of three messages for the periodic time synchronization of both DTSync and HRTS. HRTS constructs a multi-level hierarchy of the network dynamically and assigns each node with its corresponding hierarchy level or the number of hops towards the sink node, the final destination. In HRTS, a node responds to all the messages sent from nodes that have a level lower than its own level [5]. In contrast, in DTSync, a node responds only to the message sent from its reference node, which consequently results in much fewer message exchanges. This is a key advantage achieved by the proposed scheduling process. In addition, DTSync's efficient scheduling process minimizes the number of reference nodes, whereas HRTS has no scheduling process at all.

V. EXPERIMENTS WITH REAL SENSOR NETWORKS

To further validate the proposed DTSync method in practice, we implemented DTSync protocol in a C program for an embedded processor. Then we conducted experiments with real sensor networks constructed using wireless sensor hardware running the C program.

The wireless sensor board employs the NXP KW01Z128 MCU with an ARM cortex-M0 CPU core and an IEEE 802.15.4 RF transceiver [26]. A photo of the sensor board is shown in Fig. 13(a). To compare its performance with HRTS, we also implemented HRTS protocol in a C program on the same sensor board.

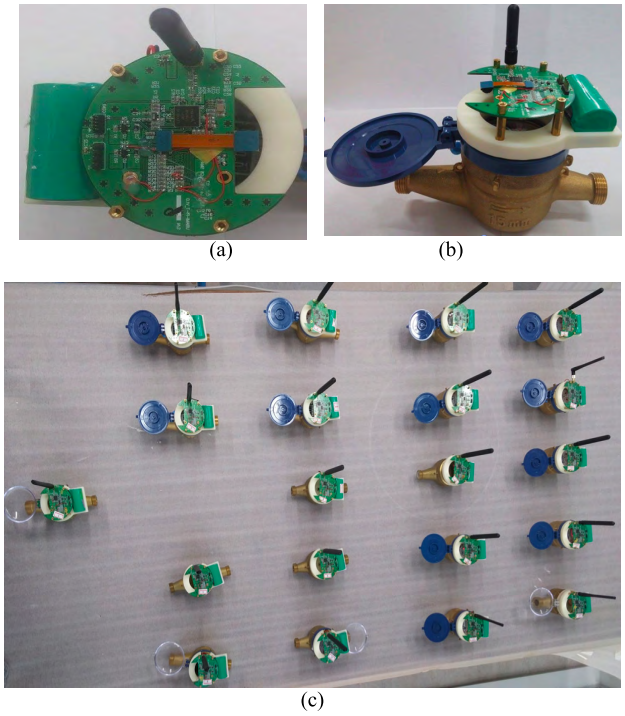

FIGURE 13. Sensor node hardware (a) Sensor board with an embedded processor (b) Sensor node with camera sensor for water meter recognition (c) A network of 20 sensor nodes for an application of wireless metering network.

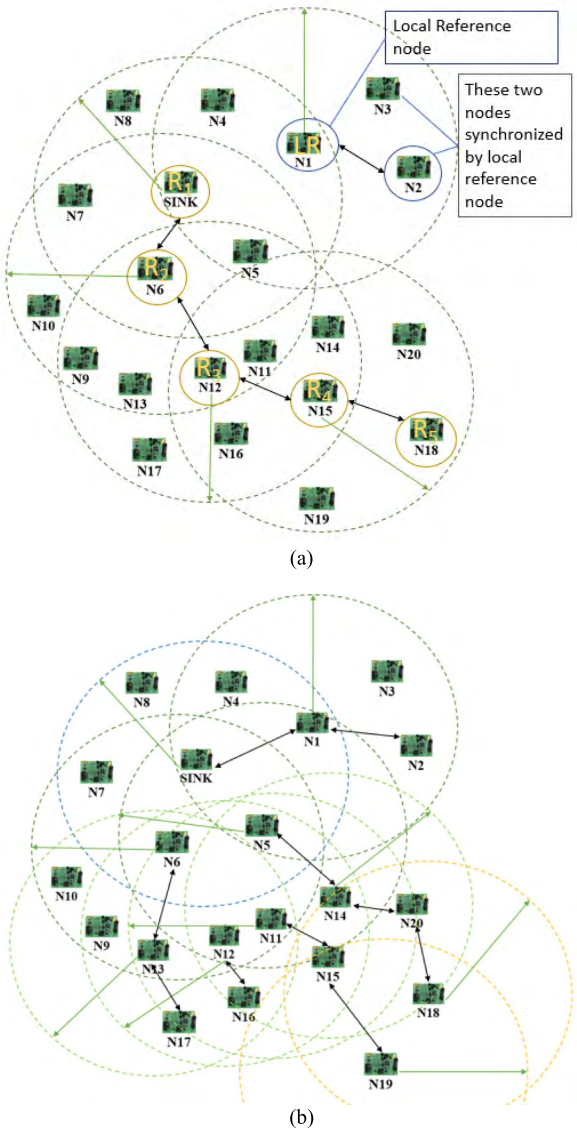

FIGURE 14. Synchronization results for the network of 20 nodes when the given wireless range forms network hierarchy of 4 levels: (a) the result of DTSync based on density, (b) the result of HRTS.

TABLE 9. Specification of sensor module.

Parameter	Value
Carrier Frequency	921.3 MHz
Channel BW	400 KHz
Bit Rate	50 Kbps
Modulation Type	FSK
TX Power	0 dBm
Frame Size	120 Byte
WTQ	20 ms
Crystal Oscillator Freq.	32768 Hz

Table 9 gives the specification of the PHY and Link layer of the sensor hardware used in the experiments. In these experiments, we applied the senor boards to a wireless water metering application with a camera to recognize the meter's numbers as illustrated by Fig. 13(b). We deployed a wireless metering sensor network of 20 nodes as shown in Fig. 13 (c). To test its operation over multi-hop networks, we used the topology illustrated in Fig. 14. This topology provides three different scenarios: two-levels, three-levels, and four-levels of synchronization. In Fig. 14, the green arrows indicate the wireless range of reference nodes, while the black arrows indicate the timing message exchanges between the current reference node and its next reference node or local reference nodes.

Here, we compare our density-based DTSync with HRTS. As demonstrated in IV.C with the simulation experiments, the number of messages is the key performance indicator.

HRTS, based on a simplistic CSMA, tends to suffer from excessive collisions and tends to leave many nodes unsynchronized [5], [6]. For the sake of fair comparison in the number of message exchanges, we ignored retransmissions incurred by the collisions during the operations of HRTS. Fig. 14 (a) shows a sequence of reference nodes elected by the density based DTSync. In the first iteration of the FTS process, the sink node (R_1) selects N6 as the next reference (R_2), since N6 has the highest density. In the second iteration, N6 (R_2) selects N12 (R_3). In the third iteration, N12 (R_3) selects N15 (R_4). In the fourth iteration, N15 (R_4) selects N18 (R_5). Then the FTS process can find no further reference nodes, although N2 and N3 still remain uncovered. Thus, the BTS process is conducted to cover the remaining nodes, N2 and N3 by selecting N1 as a local reference node. In the scheduling process of Fig. 14(a), the total number of timing messages is 15 (5 exchanges $\times$ 3 messages).

On the other hand, Fig. 14 (b) shows the operations of HRTS. In the first iteration, the sink node randomly selects node N1 among its neighbors. In the second iteration, each of N1, N5, and N6 selects N2, N14, and N13, respectively, to exchange their timing messages. In the third iteration, N11, N12, N13, and N14 choose N15, N16, N17, and N20, respectively. In the fourth iteration, N15 and N20 choose N18, and N19, respectively.

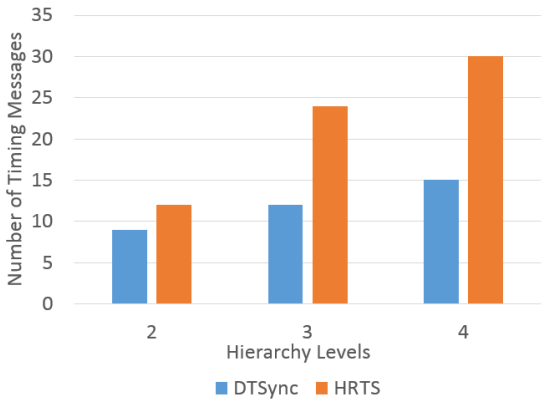

FIGURE 15. Comparison between DTSync and HRTS in the number of synchronization messages. It evaluates three different hierarchy levels for the network of Fig. 14 by configuring three different wireless ranges. The case of 2 levels is obtained using the largest wireless range. The case of 3 levels is obtained by a medium wireless range, while the case 4 levels is produced by the shortest wireless range.

The comparison of Fig. 14 (a) and (b) shows that the proposed DTSync method needs only 15 messages transmitted, whereas HRTS requires 30 messages transmitted. We also conducted similar experiments with the network of Fig. 13(c) with 2 levels and 3 levels. Fig. 15 demonstrates that DTSync outperforms HRTS by 25% fewer messages for the case of 2 levels and by 50% fewer messages for the case of 3 levels and 4 levels, respectively. Although the above real sensor network might look small, the performance of DTSync is expected to rapidly increase along with network size as proven by the simulation results of Section IV. This experiment, therefore, provides strong evidence for the key advantages of DTSync.

VI. DISCUSSION AND FUTURE WORK

DTSync is a greedy heuristic method that attempts to cover all nodes with as few reference nodes as possible through the FTS and BTS processes. There is a trade-off between the complexity of the algorithm and the optimality of the result. A global optimum algorithm would require exhaustive search, which leads to excessive complexity for large networks. In this paper, therefore, we chose a heuristic algorithm to reduce the complexity in order to cover even extremely large networks. DTSync, therefore, can be an effective synchronization method that is well suited to wireless sensor networks of various sizes.

In fact, various evaluation metrics are used in synchronization protocols such as energy consumption, accuracy and convergence time [16]. In this paper, we discussed the performance of DTSync, and addresses all of these metrics as follows.

1) **Energy consumption:** DTSync can substantially reduce the number of message exchanges, which in turn reduces the energy consumption. DTSync's objective is to reduce the power consumption during period synchronization operations rather than enhancing the accuracy of the timing. We are, therefore, primarily focused on how to reduce the number of message exchanges.
2) **Accuracy:** Nevertheless, our method does not sacrifice the accuracy of the timing. We are using the offset calculation method reported by HRTS for each periodic synchronization step, which is known to provide high timing accuracy.
3) **Convergence time:** Although DTSync can lead to a longer convergence time compared to conventional method like HRTS, it has other significant benefits. While most of conventional methods blindly select multiple reference nodes and thus incur many collisions, our scheduling process selects a minimal set of reference nodes using the notion of density metric. Its longer convergence time affects only the scheduling process, which is a preprocessing procedure conducted only once during the network initialization. We also demonstrated that the main procedure (periodic synchronization) needs only small number of hops, and is guaranteed to eliminate most of collisions.

DTSync can handle a large number of nodes with only modest overhead, and so it is scalable. Although the periodic synchronization process has very few messages, the neighbor discovery process generates a relatively large number of messages causing an increased process time. Hence, we plan to further enhance the neighbor discovery and scheduling algorithms as our future work.

VII. CONCLUSION

In this paper, we proposed a Density Table Based synchronization protocol DTSync for low power wireless

sensors network. The objectives of DTSync are (1) to reduce the energy consumption by minimizing the sequence of multi-hop synchronization steps, and (2) to guarantee the complete synchronization of all sensor nodes in the network. We evaluated the proposed DTSync algorithm through extensive simulations as well as actual implementations of wireless networks with sensor hardware. The experimental results show that DTSync achieves complete network synchronization with substantially fewer timing messages than a conventional method, HRTS. Since the number of messages poses a direct impact on the power consumption, DTSync can provide an effective solution to low power synchronization – a challenging problem, especially for large-scale multi-hop sensor networks.

REFERENCES

[1] K. S. Yıldırım and A. Kantarci, "Time synchronization based on slow-flooding in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 244–253, Jan. 2014.
[2] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. ACM 1st Int. Conf. Embedded Netw. Sensor Syst.*, 2003, pp. 138–149.
[3] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Oper. Syst. Rev.*, 2002, pp. 147–163.
[4] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proc. ACM 2nd Int. Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 39–49.
[5] H. Dai and R. Han, "TSync: A lightweight bidirectional time synchronization service for wireless sensor networks," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 8, no. 1, pp. 125–139, 2004.
[6] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. New York, NY, USA: Wiley, 2007.
[7] H. Lim, S. Kumar, and H. Kim, "Density-driven scheduling of low power synchronization for wireless sensor networks," in *Proc. IEEE Int. Conf. Electron., Inf., Commun. (ICEIC)*, Jan. 2016, pp. 1–5.
[8] C. Lenzen, P. Sommer, and R. Wattenhofer, "PulseSync: An efficient and scalable clock synchronization protocol," *IEEE ACM Trans. Netw.*, vol. 13, no. 6, pp. 717–727, Jun. 2014.
[9] C. Lenzen, P. Sommer, and R. Wattenhofer, "Optimal clock synchronization in networks," in *Proc. 7th ACM Conf. Embedded Netw. Sensor Syst.*, 2009, pp. 225–238.
[10] M. K. Maggs, S. G. O'Keefe, and D. V. Thiel, "Consensus clock synchronization for wireless sensor networks," *IEEE Sensors J.*, vol. 12, no. 6, pp. 2269–2277, Jun. 2012.
[11] J. Ammer and J. Rabaey, "Low power synchronization for wireless sensor network modems," in *Proc. IEEE Wireless Commun. Netw. Conf.*, vol. 2. Mar. 2005, pp. 670–675.
[12] F. Gong and M. L. Sichitiu, "CESP: A low-power high-accuracy time synchronization protocol," *IEEE Trans. Veh. Technol.*, vol. 65, no. 4, pp. 2387–2396, Apr. 2016.
[13] J. Wu, L. Zhang, Y. Bai, and Y. Sun, "Cluster-based consensus time synchronization for wireless sensor networks," *IEEE Sensors J.*, vol. 15, no. 3, pp. 1404–1413, Mar. 2015.
[14] J. He, P. Cheng, L. Shi, and J. Chen, "SATS: Secure average-consensus-based time synchronization in wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 61, no. 24, pp. 6387–6400, Dec. 2013.
[15] K. S. Yıldırım and Ö. Gürcan, "Efficient time synchronization in a wireless sensor network by adaptive value tracking," *IEEE Trans. Wireless Commun.*, vol. 13, no. 7, pp. 3650–3664, Jul. 2014.
[16] M. A. Sarvghadi and T.-C. Wan, "Message passing based time synchronization in wireless sensor networks: A survey," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 5, p. 1280904, 2016.
[17] J. van Greunen and J. Rabaey, "Lightweight time synchronization for sensor networks," in *Proc. 2nd ACM Int. Conf. Wireless Sensor Netw. Appl. (WSNA)*, 2003, pp. 11–19.
[18] A. D. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic gossip: Efficient averaging for sensor networks," *IEEE Trans. Signal Process.*, vol. 56, no. 3, pp. 1205–1216, Mar. 2008.
[19] J. He, P. Cheng, L. Shi, J. Chen, and Y. Sun, "Time synchronization in WSNs: A maximum-value-based consensus approach," *IEEE Trans. Autom. Control*, vol. 59, no. 3, pp. 660–675, Mar. 2014.
[20] L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization in wireless sensor network," in *Proc. 46th IEEE Decision Control*, Dec. 2007, pp. 2289–2294.
[21] M. A. A. El-Gawad, and H. Kim, "Reliable broadcast protocol based on scheduled acknowledgments for wireless sensor networks," in *Proc. Int. Conf. Electron., Inf., Commun.* 2017, pp. 485–489.
[22] I. F. Akyildiz and M. C. Vuran, *Wireless sensor networks*. vol. 4. New York, NY, USA: Wiley, 2010.
[23] H.-W. Kim and A. Kachroo, "Low power routing and channel allocation of wireless video sensor networks using wireless link utilization," *Ad Hoc Sensor Wireless Netw.*, vol. 30, pp. 83–112, Jan. 2016.
[24] S. Y. Han and D. Lee, "An adaptive hello messaging scheme for neighbor discovery in on-demand MANET routing protocols," *IEEE Commun. Lett.*, vol. 17, no. 5, pp. 1040–1043, May 2013.
[25] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu, "Z-MAC: A hybrid MAC for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 511–524, Jun. 2008.
[26] NXP, *MKW01Z128 Sub 1 GHz Low Power Transceiver Plus Microcontroller Reference Manual*, document MKW01xxRM, Rev. 2, Mar. 2014, accessed: Oct. 12, 2016. [Online]. Available: http://www.nxp.com
[27] A. N. Alvi, S. S. Naqvi, S. H. Bouk, N. Javaid, U. Qasim, and Z. A. Khan, "Evaluation of slotted CSMA/CA of IEEE 802.15.4," in *Proc. 7th Int. Conf. IEEE Broadband, Wireless Comput., Commun. Appl. (BWCCA)*, Nov. 2012, pp. 391–396.
[28] K.-L. Noh, E. Serpedin, and K. Qaraqe, "A new approach for time synchronization in wireless sensor networks: Pairwise broadcast synchronization," *IEEE Trans. Wireless Commun.*, vol. 7, no. 9, pp. 3318–3322, Sep. 2008.
[29] K. L. Noh, Q. M. Chaudhari, E. Serpedin, and B. W. Suter, "Novel clock phase offset and skew estimation using two-way timing message exchanges for wireless sensor networks," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 766–777, Apr. 2007.
[30] K.-L. Noh, Y.-C. Wu, K. Qaraqe, and B. W. Suter, "Extension of pairwise broadcast clock synchronization for multicluster sensor networks," *EURASIP J. Adv. Signal Process.*, vol. 2008, no. 1, pp. 71–80, 2008.
[31] K.-Y. Cheng, K.-S. Lui, Y.-C. Wu, and V. Tam, "A greedy distributed time synchronization algorithm for wireless sensor networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2008, pp. 2327–2331.

MAHMOUD ELSHARIEF received the B.S. degree in communications and electronics engineering from Al-Azhar University, Cairo, Egypt, in 2010, and the M.S. degree in communications and electronics engineering from Ain Shams University, Cairo, in 2014. He is currently pursuing the Ph.D. degree with the Electronics Engineering Department, Chungbuk National University, Cheongju, South Korea. From 2012 to 2016, he was a Researcher and a Teaching Assistant with the Communications and Electronics Engineering Department, Al-Azhar University. His current research focus covers the areas of wireless sensor networks and wireless vehicular communications.

MOHAMED A. ABD EL-GAWAD received the B.Sc. and M.Sc. degrees in electrical engineering from Assiut University, Assiut, Egypt, and the Arab Academy for Science and Technology, Cairo, Egypt, in 2006 and 2013, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Chungbuk National University, South Korea. From 2008 to 2015, he was with the National Telecommunication Institute, Egypt, where he conducted professional training and research on network planning. Since 2016, he has been with the MSIS Lab, Cheongju, South Korea, as a Student Researcher. His current research interests are wireless sensor networks and vehicular communications.

HYUNGWON KIM (M'95) received the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology in 1991 and 1993, respectively, and the Ph.D. degree in electrical engineering and computer science from the University of Michigan, Ann Arbor, MI, USA, in 1999. In 1999, he joined Synopsys Inc., Mountain View, CA, USA, where he developed electronic design automation software. In 2001, he joined Broadcom Corp., San Jose, CA, USA, where he developed various network chips including a Wi-Fi gateway router chip, a network processor for 3G, and 10 gigabit Ethernet chips. In 2005, he founded Xronet Corp., South Korea-based wireless chip maker, where as a CTO and CEO, he managed the company to successfully develop and commercialize wireless baseband and RF chips and software including WiMAX chips supporting IEEE802.16e and Wi-Fi chips supporting IEEE802.11a/b/g/n. Since 2013, he has been with Chungbuk National University, Cheongju, South Korea, where he is currently an Associate Professor with the Department of Electronics Engineering. His current research focus covers the areas of sensor read-out circuits, touch screen controller SoC, wireless sensor networks, wireless vehicular communications, mixed signal SoC designs for low power sensors, and bio-medical sensors.