A Distributed Delay-Efficient Data Aggregation Scheduling for Duty-Cycled WSNs

Byungseok Kang, Phan Khanh Ha Nguyen, Vyacheslav Zalyubovskiy, and Hyunseung Choo, *Member, IEEE*

***Abstract*— With the growing interest in wireless sensor networks (WSNs), minimizing network delay and maximizing sensor (node) lifetime are important challenges. Since the sensor battery is one of the most precious resources in a WSN, efficient utilization of the energy to prolong the network lifetime has been the focus of much of the research on WSNs. For that reason, many previous research efforts have tried to achieve tradeoffs in terms of network delay and energy cost for such data aggregation tasks. Recently, duty-cycling technique, i.e., periodically switching ON and OFF communication and sensing capabilities, has been considered to significantly reduce the active time of sensor nodes and thus extend network lifetime. However, this technique causes challenges for data aggregation. In this paper, we present a distributed approach, named distributed delay efficient data aggregation scheduling (DEDAS-D) to solve the aggregation-scheduling problem in duty-cycled WSNs. The analysis indicates that our solution is a better approach to solve this problem. We conduct extensive simulations to corroborate our analysis and show that DEDAS-D outperforms other distributed schemes and achieves an asymptotic performance compared with centralized scheme in terms of data aggregation delay.**

***Index Terms*— DEDAS-D, distributed aggregation scheduling, duty-cycle, WSNs, wireless sensor networks.**

I. Introduction

WIRELESS Sensor Networks (WSNs) consist of a large number of small self-powered sensing nodes, which have been used in diverse areas such as military surveillance, industrial monitoring, environment monitoring, health care, and so on to gather information or detect events [1], [2]. In many applications of WSNs, data aggregation is a fundamental operation to gather critical data from all sensor nodes to a sink node. The intermediate nodes combine all received data with their own packets into a single packet by some aggregation function such as SUM, MAX, MIN, AVERAGE, and so on [3], [4].

Minimizing the data aggregation delay, i.e., the time for aggregated data to reach the sink, is important for applications that are required to take actions based on deadlines, such as real-time monitoring or mission-critical applications [5]–[7]. A Minimum-Latency Aggregation Scheduling (MLAS) problem that finds the fastest collision-free schedule for data aggregation subject to interference constraints has been studied in previous literature. Interference is one of the crucial factors that affects the overall performance of WSNs. When a node hears the signals from more than one transmitter at the same time, a collision occurs due to signal interference. The node cannot receive successfully any data packet, so the transmitter should retransmit data packets. Collision [8], [9] is a challenge in WSNs because the packet retransmission wastes a large amount of energy. Solving the collision issue in the MAC layer (hardware) rather than in the application layer (scheduling algorithm) can result in an increase of delay in the data aggregation process. Therefore, many works address MLAS problem by scheduling the data aggregation operation of each node based on the network topology such that no collision happens and the time for all data to be aggregated to the sink is minimized.

In addition, minimizing energy consumption [10]–[12] is a fundamental challenge to operate WSNs for a long duration since sensor nodes are typically powered by non-rechargeable batteries with limited capacity. Recently, a duty-cycling technique [13], [14], where each sensor node works for a short duration and sleeps for the remaining time in each working period, has been studied. This technique is an efficient solution to save a large amount of energy because it can significantly decrease the active time of sensor nodes. However, sleep latency [15], i.e., a certain period which a sender has to wait until its receiver becomes active, due to the duty cycle, results in a substantial degradation of network delay. Therefore, the MLAS problem in duty-cycled WSNs has emerged as a new problem.

In this paper, we focus on designing a distributed scheme to solve the MLAS problem in duty-cycled WSNs. The main contributions of our paper are four-fold.

- We present a formal equation for the MLAS problem in duty-cycled WSNs and analyze the drawbacks of existing schemes solving this problem.
- We design a Distributed Delay-Efficient Data Aggregation Scheduling (DEDAS-D) scheme to generate a collision-free aggregation schedule and minimize the data aggregation delay. DEDAS-D contains two novel ideas in the tree construction phase and the scheduling phase. To reduce the effect of high degree nodes on the data aggregation delay, we propose a distributed algorithm to construct a Degree-Constrained data aggregation

Manuscript received January 12, 2017; revised February 20, 2017, March 27, 2017, and March 28, 2017; accepted April 5, 2017. Date of publication April 7, 2017; date of current version May 5, 2017. This work was supported in part by the G-ITRC Program under Grant IITP-2015R6812-15-0001, in part by the NRF Research Fellow Program under Grant NRF-2016R1A6A3A11934080, and in part by the NRF Korea under Grant 2010-0020210. The associate editor coordinating the review of this paper and approving it for publication was Dr. Rosario Morello. *(Corresponding author: Hyunseung Choo.)*

B. Kang, P. K. H. Nguyen, and H. Choo are with the College of Information and Communication Engineering, Sungkyunkwan University, Suwon 16419, South Korea (e-mail: fbyungseok@skku.edu; npkha@skku.edu; choog@skku.edu).

V. Zalyubovskiy is with the Laboratory of Discrete Optimization in Operations Research, Sobolev Institute of Mathematics, 630090 Novosibirsk, Russia (e-mail: slava@math.nsc.ru).

Digital Object Identifier 10.1109/JSEN.2017.2692246

Tree (DCT) with a predefined degree threshold. In addition, a fast scheduling algorithm is used in the scheduling phase to generate a collision-free schedule and minimize the data aggregation delay. To the best of our knowledge, our paper is the first work that proposes a distributed approach to solve the MLAS problem in duty-cycled WSNs. A duty-cycle is the fraction of one period in which a signal is active. By using this technique, we can have reduced energy consumption for every sensor.

- We prove the correctness of our proposed scheme and show that DEDAS-D is a better solution for this problem compared with previous schemes.
- We conduct extensive simulations to verify our analysis and evaluate the effectiveness of our proposed scheme. The simulation results show that both the novel idea in DEDAS-D significantly reduce the data aggregation delay. Comparing to a related distributed work solving the MLAS problem [16] and the first centralized scheme solving the MLAS problem in duty-cycled WSNs [17], DEDAS-D improves the delay performance by 65.04% and 50.95%, respectively. Moreover, it is worth noting that the delay performance of DEDAS-D approximates to that of our previously proposed centralized approach [18].

The rest of this paper is organized as follows. We first review the related work in Section II. In Section III, we describe necessary assumptions, system models, and formally define the problem. Our proposed scheme is shown in Section IV. We provide the analysis in Section V and then show simulation results in Section VI. Finally, the conclusion is presented in Section VII.

II. Related Work

Since data aggregation is an essential operation in sensor networks, extensive research has been conducted on data aggregation with different targets such as minimizing energy consumption, maximizing fairness, or minimizing latency [5], [19], [20]. Among these targets, minimizing the latency of data aggregation is the most popular problem that is considered in the papers.

The MLAS problem is to design the transmission schedule for all sensors such that there is no conflict between any two concurrent transmissions, and minimize the number of time slots for all data to reach the sink. The MLAS problem is proved Non-deterministic Polynomial-time hard (NP-hard) and many approximation algorithms have been proposed to solve this problem. In [21], the authors present an algorithm named Shortest Data Aggregation (SDA) with the latency bound $(\Delta - 1)R$, where Δ is the maximum node degree in a network graph and R is the network radius. Huang *et al.* [22] propose the First-Fit Scheduling (FFS) algorithm based on the maximal independent set (MIS) and prove that their algorithm has the smaller latency bound $23R + \Delta - 18$ compared with SDA. In [23], Wan *et al.* propose the Enhanced Pipelined Aggregation Scheduling (E-PAS) algorithm with the aggregation schedule latency is at most $(1 + \mathcal{O}(\frac{log(R)}{\sqrt[3]{R}}))R + \Delta$. The main idea of E-PAS algorithm is similar to that of FFS algorithm. However, instead of being fixed, parent nodes are dynamically determined during the scheduling process based on the minimum cover set.

Along with centralized algorithms, many distributed approaches have been studied to solve the MLAS problem. Yu *et al.* [24] propose a first distributed scheduling algorithm, called Distributed Aggregation Scheduling (DAS), generating a collision-free schedule with the latency bound of $24D+6\Delta+16$, where D is the network diameter. In [16], Xu *et al.* design centralized and distributed Improved Aggregation Scheduling (IAS) algorithms using the protocol interference model with an upper bound on delay of $16R + \Delta - 14$. They only theoretically prove as $16R + \Delta - 14$ is the most value, this is a theoretical limit. They also prove that the overall lower delay bound of IAS in any interference model is $max\{R, log_n\}$, where n is the number of nodes in the network. Both DAS and IAS contain two phases. In the first phase, a data aggregation tree is built by finding the connected dominating set (CDS) as a virtual backbone of sensor network. Then in the second phase, a TDMA schedule of nodes is constructed. In [25], an energy-efficient distributed scheduling algorithm called Clu-DDAS is proposed. This algorithm constructs a cluster-based data aggregation tree which is different from the MIS/CDS based aggregation tree used in [16], [22], and [24]. However, no aforementioned schemes are appropriate for duty-cycled WSNs since the operation of nodes in such networks is different from that of conventional WSNs.

Recently, the duty-cycling technique has emerged as an efficient solution to conserve the sensors' energy and prolong network lifetime. In duty-cycled WSNs, sensor nodes are scheduled to be in an active state for only a short period and then stay in a sleep state for a long time. However, such duty-cycled WSNs undergo many challenges. In particular, sleep latency introduced in scheduled-based duty-cycled WSNs causes a significant increase of end-to-end delivery delay. Meanwhile, many applications require a delivery time bounded by a low specific deadline. Therefore, many works focus on solving this problem. In [15], Yu *et al.* propose a novel dynamic switch-based forwarding technique for low duty-cycled WSNs with unreliable communication links to achieve optimal expected data delivery ratio, expected communication delay, or expected energy consumption. The main idea is that each node maintains multiple potential forwarding nodes at each hop to reduce the effect of sleep latency brought by the duty cycle. The authors in [26] introduce three different approaches to guarantee a real-time communication delay: an active bits augmentation scheme, a sink augmentation scheme, and a hybrid scheme. Guo *et al.* [27] propose an energy-efficient flooding design for low duty-cycled WSNs by adopting a novel flooding tree, which allows nodes of common parents to wake up simultaneously. Additionally, many algorithms [28]–[31] are proposed to solve the broadcasting problem in duty-cycled WSNs with different targets such as minimizing broadcasting schedule latency, minimizing the number of broadcasting transmissions, minimizing the energy broadcast, and so on.

Although there are many studies on duty-cycled WSNs, most of them focus on solving problems related to data forwarding, data flooding, and broadcasting. Paper [17] is

the first work on MLAS problem in duty-cycled WSNs. The authors show that this problem is NP-hard and propose an approximation Scheduling Algorithm (SA) to address this problem. The SA contains two phases: layered structure construction and working period scheduling. In the first phase, a layered structure for aggregation is generated using the Maximal Independent Set. In the second phase, all nodes in the network are scheduled layer by layer using a sub-procedure minimal covering (MC) schedule. However, the SA has the following disadvantages that prevent it from achieving a better data aggregation delay.

- First, the MC schedule of the SA only considers the active time slot of a receiver and does not consider sleep latency between the sender and receiver. Nevertheless, as mentioned previously, sleep latency is the main factor that increases the data aggregation delay. As a result, the SA cannot generate a good schedule.
- Second, to generate a collision-free schedule, in the working period scheduling phase, the SA delays the working period of scheduled nodes in layer i by the latest scheduled working period of nodes in all deeper layers of layer i. This collision-avoidance method causes high data aggregation delay. The reason is that nodes in layer i can be scheduled to transmit their data in the same working period of nodes in deeper layers as long as they are not conflicted.

In our previous work [18], we propose a centralized scheme called DEDAS-B to solve the MLAS problem in duty-cycled WSNs. DEDAS-B achieves a lower data aggregation delay compared with SA by constructing a balanced shortest path tree as a data aggregation tree and utilizing a novel collision-avoidance method to generate a collision-free aggregation schedule.

In summary, the application of previous schemes solving the MLAS problem in conventional WSNs to duty-cycled WSNs cannot achieve a good delay performance because of the differences in their characteristics. Due to their considerations for the specific properties of duty-cycling environment, SA and DEDAS-B are more suitable solutions than other previous schemes to tackle the MLAS problem in duty-cycled WSNs. However, both of them are centralized schemes. From the viewpoint of practical implementation, a centralized approach is not efficient because the sink has to collect the information of the entire network. Moreover, the schedule information should be broadcast to all nodes in the network; thus, it consumes a lot of energy. This motivates our work to propose a distributed approach to solve the MLAS problem in duty-cycled WSNs. In this paper, we investigate the MLAS problem in duty-cycled WSNs and propose a novel distributed scheme to significantly reduce the data aggregation delay.

III. Preliminaries

A. Network Model

We model the sensor network as a unit-disk graph $G = (V, E)$, where V is the set of sensor nodes and E is the set of edges in the network. All nodes are located in a Euclidean plane and equipped with an omni-directional antenna. For the sake of simplicity, we assume that the transmission range of all nodes, denoted by R, is fixed and the same. An edge $(u, v) \in E$ if and only if $\|u - v\| \leq R$, where $\|u - v\|$ is the Euclidean distance between u and v.

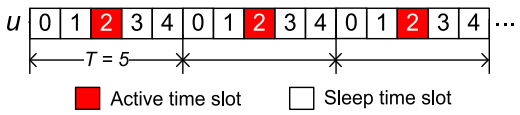


Fig. 1. Working schedule of a sensor in duty-cycled WSNs.

In duty-cycled WSNs, the entire lifetime of each node is divided into multiple working periods with the same length (T time slots) such that the duration of each time slot is long enough for a sensor to send and receive one data packet. Each node u randomly selects one of T time slots as its active time slot, denoted by $A(u)$, and sleeps in the remaining $T - 1$ time slots in a working period (see Fig. 1). The duty cycle is defined as $1/T$. Fig. 1 demonstrates the working schedule of a sensor node u in duty-cycled WSNs with the active time slot is 2 and its duty-cycle is 20% ($T = 5$).

We assume that each node has only one data packet to send. Each node can only receive data in its active time slot but can activate its communication module for sending the data packet at any time slot. A node cannot send and receive data at the same time slot. We also assume that a node is able to aggregate data perfectly from all its child nodes and send to its parent node in the data aggregation tree, i.e., two or more data packets can be aggregated fully into a new equal-sized packet. A sink $s \in V$ collects data from all other nodes. The aggregation process is complete when all data packets in the network reach the sink.

B. Delay Model

We define two following variables, which describe the data aggregation delay of a sensor node.

Definition 1 (Sleep Latency $sl(u, v)$): Sleep latency for the transmission from a sender u to a receiver v, denoted by $sl(u, v)$, is the time sender u spends waiting for its receiver v to wake up brought by the duty-cycling mode. This sleep latency is calculated as follows:

$$sl(u, v) = \begin{cases} A(v) - A(u) & \text{if } A(v) > A(u) \\ A(v) + T - A(u) & \text{if } A(v) \leq A(u) \end{cases} \quad (1)$$

where T is the length of a working period; $A(u)$ and $A(v)$ are the active time slot of u and v, respectively.

Definition 2 (E2E Delay $d(u)$): The end-to-end (E2E) delay of node u, denoted by $d(u)$, is the time for a data packet sent by u to be received by the sink s if interferences (primary and secondary signal interferences) are ignored. This delay can be considered as the lower bound of data aggregation delay of u.

$$\begin{cases} d(s) = 0 \\ d(u) = d(z) + sl(u, z) \end{cases} \quad (2)$$

where z is the parent node of u in the data aggregation tree.

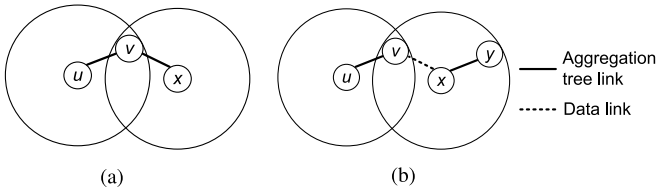


Fig. 2. Two types of interference in duty-cycled WSNs. (a) Primary interference. (b) Secondary interference.

C. Interference Model

Jain *et al.* [32] present a protocol interference model. We adopt this protocol interference model with some modifications to adapt it to a duty-cycling environment. We assume that a node cannot send and receive data at the same time slot. Let R and R' be the transmission range and interference range of node u, $R \leq R'$. For the sake of simplicity, we assume that $R = R'$ and there is a single wireless channel. A transmission from a sender u to its receiver $p(u)$ is successful if both of the following conditions are satisfied:

1) $\|u - p(u)\| \leq R$.
2) Any node x such that $\|x - p(u)\| \leq R$ and $A(p(x)) = A(p(u))$ is not transmitting.

Based on the protocol interference model, we classify the signal interference into two types: primary and secondary interference [33]. Fig. 2 shows two types of interference causing collision in duty-cycled WSNs. In Fig. 2(a), primary interference occurs when u and x are scheduled to send data to v in the same working period. Hence, v will not receive any data from u and x. Secondary interference occurs when a node within the transmission range of other nodes is scheduled to send (receive) a packet in the same working period with its unintended receivers (senders). For example, in Fig. 2(b), assume that $A(v) = A(y)$ and there are two ongoing scheduled transmissions from u to v and from x to y in the same working period. Because v is in the transmission range of its unintended sender x, v will not receive successfully data from its intended sender u.

To solve the collision issue and guarantee that a collision-free aggregation schedule is generated, we give the definition of the conflicting set for each sensor node. Each sensor node should keep this information in the data aggregation scheduling process. When scheduling, a sensor node should check to ensure that its scheduled transmission is not conflicted with any assigned transmission of node in its conflicting set. The collision is prevented by this method.

Definition 3 (Conflicting Set $CS(u)$): One node is called a conflicting node of u if it cannot transmit data concurrently with u to prevent the collision. The set of conflicting nodes of node u is defined as u's conflicting set. $CS(u)$ consists of:

1) All child nodes of u's parent node $p(u)$ (except node u).
2) All child nodes of a neighboring node of u, denoted by v, if $A(v) = A(p(u))$ (except all descendants and ascendants of u in the data aggregation tree).
3) Neighboring node of $p(u)$, denoted by x, if $A(p(x)) = A(p(u))$ (except all descendants and ascendants of u in the data aggregation tree).

Suppose that u is sending data to $p(u)$ and the collision occurs. In the first case, the transmission of u collides with that of its sibling nodes due to primary interference. In the second case, the collision occurs at a neighboring node of u because of the secondary interference between u and the child nodes of this neighboring node. In the last case, the collision occurs at $p(u)$ due to the secondary interference between u and a neighboring node of $p(u)$. We note that all descendants and ascendants of u in the data aggregation tree are excluded from its conflicting set since a node cannot send data until receiving data from all its descendants. The conflicting node and conflicting set information of each node can be obtained after the data aggregation tree is constructed by broadcasting.

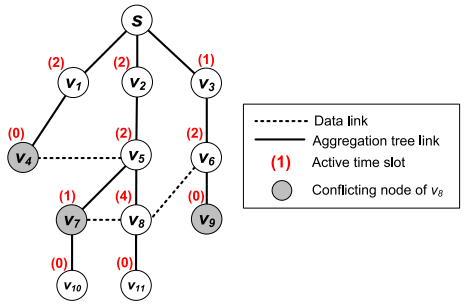


Fig. 3. An example of conflicting set.

Fig. 3 shows an example of the conflicting set of a sensor node. Given a data aggregation tree with 12 nodes, let us consider node v_8. Nodes v_7, v_9, and v_4 are included in $CS(v_8)$ following case 1, case 2, and case 3 of Definition 3, respectively.

D. Problem Formulation

We formulate the Minimum-Latency Aggregation Scheduling (MLAS) problem in duty-cycled WSNs, which is considered in [17] as follows. Given a network graph $G = (V, E)$ and a sink node $s \in V$, let S_i be the set of scheduled senders sending data in working period i such that any pair of transmissions of senders in S_i are not conflicting. A data aggregation schedule is defined as $S = \bigcup_{i=1}^{L} S_i$, where L is the data aggregation delay. The MLAS problem in duty-cycled WSNs is formally defined as:

Finding a data aggregation schedule S which minimizes L s.t.

1) $S = \bigcup_{i=1}^{L} S_i = V \setminus \{s\}$.
2) $S_i \bigcap S_j = \emptyset \ \forall i \neq j$.
3) All data are aggregated to the sink s in L working periods.

In this paper, the schedule of a sensor node u is denoted as $< t_{sch}(u), p(u) >$, which means u is scheduled to send data to its parent $p(u)$ in $A(p(u))$ of working period $t_{sch}(u)$. We have to find the collision-free aggregation schedule for all nodes in the network that minimizes the data aggregation delay, i.e., minimizes the required number of working periods for the sink to receive all data packets in the network.

IV. Proposed Scheme

Our main contribution in this paper is a Distributed Delay-Efficient Data Aggregation Scheduling (DEDAS-D) scheme to solve the MLAS problem in duty-cycled WSNs.

DEDAS-D contains two phases: 1) data aggregation tree construction and 2)data aggregation scheduling. In the first phase, we build a degree-constrained aggregation tree (DCT) by a distributed algorithm to mitigate the impact of high degree nodes on data aggregation delay. Given a predefined degree threshold α, a data aggregation tree where the number of child nodes of each node is not greater than α is constructed. This α is mandatory parameter in our environments. However, in the cases where the degree constraint of all nodes cannot be satisfied (e.g, when the network density is not high enough), the proposed algorithm tries to minimize the number of nodes having the degree greater than α. After the DCT is built, in the second phase, each node in the network is scheduled to send data to its parent node such that its transmission schedule does not interfere with other transmission schedules by a Fast Data Aggregation Scheduling algorithm.

A. Aggregation Tree Construction Algorithm

To minimize the data aggregation delay, a data aggregation tree should maximize the number of nodes transmitting in the same working period without collisions. In this section, first we investigate the relationship between the number of high degree nodes in the data aggregation tree and the data aggregation delay. We point out that the increase of the number of high degree nodes reduces the number of nodes transmitting simultaneously without collisions and thus increases the data aggregation delay. Based on this observation, a distributed DCT algorithm is proposed to alleviate the effect of high degree nodes on data aggregation delay.

We start with a Theorem indicating the lower delay bound of an aggregation schedule with a given data aggregation tree.

Theorem 1: Given a data aggregation tree, for any collision-free aggregation schedule, the data aggregation delay L in duty-cycled WSNs cannot be smaller than $\xi_{max} = max\{\xi_i \mid i = 1, ..., N\}$, where ξ_i is the number of child nodes of node i in the aggregation tree.

Proof: We prove Theorem 1 by contradiction. We assume that $L < \xi_{max}$ and let k be the node that has ξ_{max} child nodes. This means at least two child nodes of k have to transmit in the same working period that is impossible to guarantee a collision-free schedule. Therefore, the assumption $L < \xi_{max}$ is incorrect. ■

Theorem 1 shows that the delay bound of an aggregation schedule with a given data aggregation tree depends on the maximum number of child nodes of a node. A tree having the small value of ξ_{max} has a higher probability to result in a lower delay. Therefore, reducing the number of high degree nodes in the data aggregation tree is important to minimize the data aggregation delay. This motivates us to construct a DCT that restricts the degree of all nodes in the tree based on a predefined degree threshold α. Since finding the degree-constrained spanning tree problem is proved NP-complete [34], we propose an approximation algorithm to build a DCT in a distributed manner.

First, we introduce some symbols used in our proposed DCT algorithm.

- $n_n(u)$: number of neighbors of node u.
- $ChS(u)$: set of child nodes of u in the DCT.
- $n_{ch}(u)$: number of child nodes of u in the DCT.
- $n_{apc}(u)$: number of available parent candidates (APC) of u. A neighboring node v is called an APC of u if v satisfies both conditions: 1) v has not been selected as a child node of u in the DCT and 2) $n_{ch}(v) < \alpha$.
- $NAS(u)$: set of selected activators of u. The selected activators of u are u's child nodes that continue constructing the DCT after being added to the DCT.
- $state(u)$: state of u. Node u can be in one of the following four states: WAIT, ACTIVATED, LISTEN, and COMPLETE.

Additionally, all nodes in the network are categorized into two types based on the value of n_{apc}:

- *Special node*: one node u is defined as special node if $n_{apc}(u) = 1$.
- *Normal node*: one node u is defined as normal node if $n_{apc}(u) > 1$.

DCT algorithm contains two stages: the start-up stage and the main stage. In the first stage, special nodes select their unique neighbors as parent nodes. This stage is completed when no special node exists in the network. In the second stage, the sink s becomes the first activator. An activator selects the child nodes from its neighbors. Among those child nodes, new activators are selected to continue building the DCT. If an activator cannot find any new activator, it will notify its parent in the DCT to find another new activator. Once the sink and all its selected activators cannot find any new activator, the algorithm ends.

Algorithm 1: Start-up Stage of DCT Algorithm

Initialization: $d(u) = 0, n_{ch}(u) = 0, n_{apc}(u) = n_n(u)$, $NAS(u) = \emptyset, ChS(u) = \emptyset$, $state(u) =$ WAIT $\forall u \in V$

1: **while** $\exists v \in V, n_{apc}(v) = 1$ **do**
2: v sends a JOIN_REQ message to its one-hop neighbors
3: **if** u receives JOIN_REQ from v **then**
4: $n_{ch}(u) = n_{ch}(u) + 1$; $n_{apc}(u) = n_{apc}(u) - 1$
5: u sends a JOIN_ACCEPT message to its one-hop neighbors
6: **end if**
7: **if** v receives JOIN_ACCEPT from u **then**
8: $p(u) = v$; $state(u) =$ COMPLETE;
9: **end if**
10: Other neighbors of u decrease their n_{apc} by one if $n_{ch}(u) \geq \alpha$
11: **end while**

We next discuss two stages of the DCT algorithm in more detail. Algorithm 1 describes the start-up stage of the DCT algorithm. Initially, all nodes are in WAIT state. In the start-up stage, each special node v ($n_{apc}(v) = 1$) sends a JOIN_REQ message containing v's ID to its neighbor to request to join the DCT. Upon receiving the JOIN_REQ message from v, its unique neighbor u updates its local variables and broadcasts a JOIN_ACCEPT message (lines 3-6). v joins the DCT and changes its state to COMPLETE when receiving JOIN_ACCEPT message (lines 7-9). Meanwhile,

other neighbors of u decrease their numbers of available parent candidates by one if $n_{ch}(u) \geq \alpha$ because u is no longer an APC of those nodes. It should be noted that after updating its n_{apc} (line 4 or line 10), a normal node can become a new special node and the above-mentioned process is repeated.

The sink s can detect that the start-up stage has been completed by a simple acknowledgement procedure as follows. First, a Breadth First Search (BFS) tree is constructed by a distributed algorithm shown in [35]. Then each special node after joining the DCT or normal node will send an ACK message to the parent if and only if it has already received ACK messages from all its child nodes in the BFS tree. When s receives ACK messages from all its child nodes in the BFS tree, it concludes that the start-up stage has been finished.

The pseudocode of the main-stage of the DCT algorithm is shown in Algorithm 2. In the main stage, if all neighbors of the sink s have already been selected as the child nodes of s in the start-up stage, i.e., the DCT has already been constructed, the DCT algorithm terminates (line 2). Otherwise, s becomes the first activator and starts finding its child nodes. When $state(s) \neq$ COMPLETE, the algorithm works as follows.

An activator u sends a FIND_CHILD message containing its ID to the neighbors to find child nodes. After receiving JOIN_REQ messages from its neighbors in WAIT state, u sets a backoff time to zero for each special neighbor (line 13). For each u's normal neighbor v, its backoff time $t(u, v)$ is set based on its E2E delay with u as the parent node, its number of current child nodes, and a random number in the interval between zero and one (line 15). Therefore, $t(u, v)$ is proportional to the delay for data transmitted from v to the sink s via u.

For each neighbor v, during its backoff time, if u receives a JOIN_CONFIRM message of v for another activator, i.e., v has already selected that activator as its parent node since the transmission via that activator gets the smaller delay, u cancels the backoff time $t(u, v)$ (line 18). Otherwise, after the backoff time $t(u, v)$ has expired, u executes *Procedure 1* (line 20). Since the backoff time of special nodes is zero, special neighbors are always selected as child nodes of u. For each u's normal neighbor v, once $t(u, v)$ has expired, u selects v as its child node and adds this node to the set $NAS(u)$ as a new activator (line 4, *Procedure 1*).

After a new child node is selected from u's special or normal neighbors, u updates $n_{ch}(u)$ and sends a JOIN_ACCEPT message containing selected child node's ID to its one-hop neighbors (lines 1-2, *Procedure 1*). To guarantee the degree limitation, u compares the value of $n_{ch}(u)$ with that of the degree threshold α. If a new activator has already been selected and $n_{ch}(u) = \alpha$, u stops its child node selection process and changes the state to LISTEN (lines 5-9, *Procedure 1*). In this case, we guarantee that the degree of u is not greater than α. However, if u has not selected any new activator yet, it continues its child node selection process. Since the tree construction cannot proceed unless a new activator has already been selected, in this case, we still allow u to add additional child nodes, even if this results in a degree constraint violation of u (line 11, *Procedure 1*).

Algorithm 2: Main Stage of DCT Algorithm

```
1: if sink s has n_ch(s) = n_n(s) then
2:     return
3: else
4:     s becomes the first activator: state(s) = ACTIVATED
5:     while state(s) ≠ COMPLETE do
6:         for each activator u (state(u) = ACTIVATED) do
7:             u sends a FIND_CHILD message to its one-
               hop neighbors
8:             for each neighbor v of u in WAIT state, upon
                   receiving FIND_CHILD do
9:                 v sends JOIN_REQ to u
10:            end for
11:            for each JOIN_REQ received from v do
12:                if n_apc = 1 then
13:                    t(u, v) = 0
14:                else
15:                    t(u, v) = d(u)+sl(v, u)+rand(0, 1)+
                       (n_ch(v) * T)
16:                end if
17:                if u receives JOIN_CONFIRM of v for another
                       activator z when t(u, v) has not expired then
18:                    u cancels t(u, v)
19:                else
20:                    u executes Procedure 1
21:                end if
22:            end for
23:            if u does not find any new activator then
24:                u sends a STUCK message to p(u);
                   state(u) = COMPLETE
25:            end if
26:        end for
27:        if v receives JOIN_ACCEPT from u then
28:            v executes Procedure 2
29:        end if
30:        for each neighbor y of u in WAIT state (except v),
               upon receiving JOIN_ACCEPT of u for v do
31:            if n_ch(u) ≥ α then
32:                n_apc(y) = n_apc(y) − 1;
33:            end if
34:        end for
35:        for each node u in LISTEN state or sink s, upon
               receiving STUCK from all nodes in its NAS do
36:            u executes Procedure 3
37:        end for
38:    end while
39: end if
```

In case u cannot find any new activator, it sends a STUCK message to the parent node in DCT to request this node to find another new activator and then changes its state to COMPLETE (lines 23-25). Upon receiving a JOIN_ACCEPT message from an activator u, one node v in the WAIT state will join the DCT. *Procedure 2* shows the operation of v when joining the tree. v sends a JOIN_CONFIRM message to its one-hop neighbors containing its ID and the parent node's ID to notify that it has already selected the parent node in DCT.

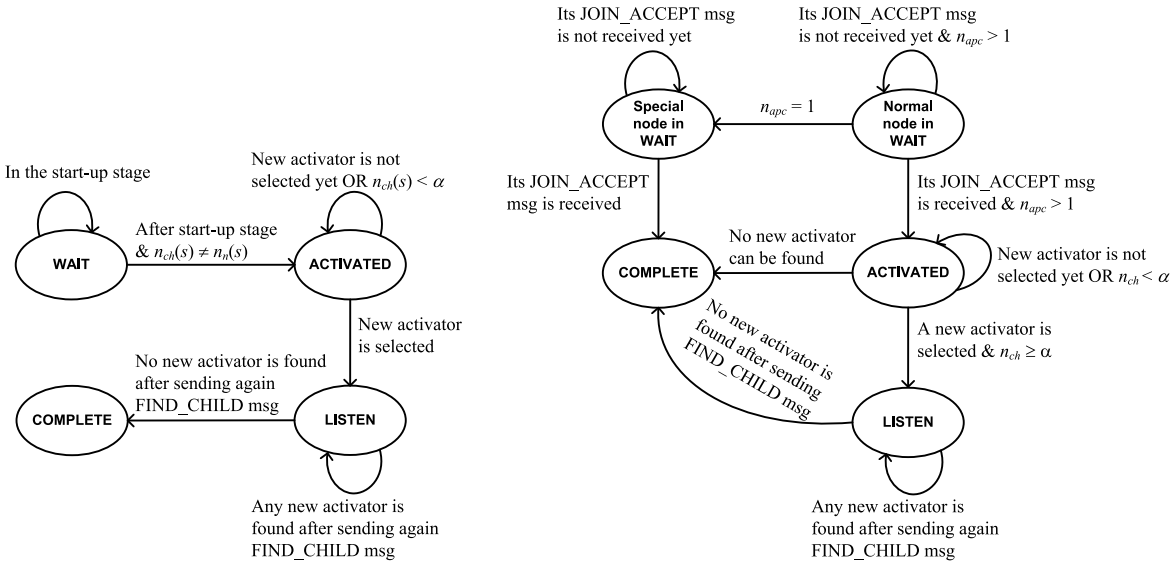


Fig. 4. State diagram of DCT Algorithm. (a) Sink. (b) Other nodes

Procedure 1: Selecting Child Nodes & New Activators

//Operation of activator u when $t(u, v)$ has expired

1: $ChS(u) = ChS(u) \cup v$; $n_{ch}(u) = n_{ch}(u) + 1$
2: u sends a JOIN_ACCEPT to its one-hop neighbors
3: **if** $n_{apc}(v) \neq 1$ **then**
4: $NAS(u) = NAS(u) \cup v$
5: **if** $n_{ch}(u) < \alpha$ **then**
6: u continues its backoff times for other neighbors
7: **else**
8: u cancels its backoff times for other neighbors; $state(u) =$ LISTEN
9: **end if**
10: **else**
11: u continues its backoff times for other neighbors
12: **end if**

Procedure 2: Joining DCT

//Operation of node v in WAIT state when receiving its JOIN_ACCEPT message sent from u

1: $p(v) = u$; $d(v) = d(u) + sl(v, u)$
2: v sends a JOIN_CONFIRM message to its one-hop neighbors
3: **if** $n_{apc}(v) = 1$ **then**
4: $state(v) =$ COMPLETE
5: **else**
6: $state(v) =$ ACTIVATED
7: **end if**

If v is a special node, since it does not have any available link with neighbors, v changes its state to COMPLETE and cannot participate in the tree construction any more. Otherwise, if v is a normal node, it becomes a new activator and continues constructing the DCT.

Procedure 3: Finding New Activators

//Operation of node u when receiving STUCK messages from all nodes in its NAS

1: u sends FIND_CHILD again to its one-hop neighbors
2: **if** u does not receive any JOIN_REQ from normal node **then**
3: u sends STUCK to $p(u)$; $state(u) =$ COMPLETE
4: **end if**
5: **if** sink s does not receive any JOIN_REQ from normal node **then**
6: $state(s) =$ COMPLETE
7: **end if**

After receiving STUCK messages from all nodes in the NAS set, a node in LISTEN state knows that all its selected activators cannot find any new activator (line 35). Therefore, it executes *Procedure 3* to find new activators. This node rebroadcasts a FIND_CHILD message to its one-hop neighbors. If no activator can be found, it continues requesting the parent to find a new activator by sending a STUCK message. When the sink s receives STUCK messages from all nodes in $NAS(s)$ and cannot find any new activator, s changes its state to COMPLETE and the DCT algorithm ends. Fig. 4 summarizes the states of nodes in the DCT algorithm.

B. Fast Distributed Aggregation Scheduling Algorithm

In this section, we first present the main idea of our proposed Fast Distributed Aggregation Scheduling (FDAS) algorithm and then explain this algorithm in detail. Finally, we compare the FDAS algorithm with the Improved Aggregation Scheduling (IAS) algorithm [16], which is one of the most recent distributed data aggregation scheduling algorithms.

1) Algorithm Description: A schedule for node u in duty-cycled WSNs is a working period for u to transmit data to its parent node $p(u)$ in $A(p(u))$ of this working period. The second phase of DEDAS-D is a Fast Data Aggregation Scheduling (FDAS) algorithm to find the schedules for all nodes in a distributed manner such that the scheduled transmission of each node does not conflict with that of other nodes.

To execute the FDAS algorithm, each node u should store the following local variables.

- $rank(u)$: rank of u, which is a couple of $level(u)$ and u's ID. $level(u)$ is defined as the hop distance from u to the sink s in the DCT. We say $rank(u) > rank(v)$ if 1) $level(u) > level(v)$ or 2) $level(u) = level(v)$ and node u's ID $>$ node v's ID. Each node can collect this information through a broadcasting process from the sink.
- $t_{sch}(u)$: scheduled working period for the transmission of u, which is the assigned working period for u to transmit data to its parent node $p(u)$.
- $t_{min}(u)$: minimum valid working period for u's transmission, which is defined as the maximum scheduled working period of u's child nodes. We have $t_{sch}(u) \geq t_{min}(u)\ \forall u \in V \setminus \{s\}$.
- $n_{uch}(u)$: number of unscheduled child nodes of u.
- $CS(u)$: conflicting set of u, which is defined in Definition 3.
- $FS(u)$: set of forbidden working periods of u. $FS(u)$ indicates the set of working periods which u should not select for its transmission to prevent the collisions, i.e., set of scheduled working periods of u's conflicting nodes.

The main idea of the FDAS algorithm is as follows. An unscheduled node whose all descendants in the DCT have already been scheduled (called ready node) competes with the other ready nodes to be scheduled. The node with the highest rank among competitors has the highest priority and selects an appropriate working period for its transmission first. To prevent the collisions, this selected working period should be different from all working periods that have already been assigned to nodes in its conflicting set. After finishing the scheduling process, the node broadcasts a notification message to its neighbors. Upon receiving this message, other nodes update its local variables.

Algorithm 3 describes the pseudocode of the FDAS algorithm. First, each node in the network (except the sink s) initializes its local variables. A ready node u which has the highest rank compared with all its conflicting nodes is scheduled first. If u is a child node in DCT, its scheduled working period $t_{sch}(u)$ is assigned to one (line 4). Otherwise, if all its child nodes of u have already finished the scheduling process, the scheduled working period for u is calculated based on the active time slots of u and $p(u)$, and the maximum scheduled working period of u's child nodes $t_{min}(u)$. In case $A(u) < A(p(u))$, u can transmit data to the parent node $p(u)$ in the same working period $t_{min}(u)$ which it received data from its child node (line 7). Otherwise, if $A(u) \geq A(p(u))$, u should wait for the next working period of $p(u)$ to transmit its received data (line 9). To generate a collision-free aggregation schedule, u should guarantee that the selected $t_{sch}(u)$ is different from the scheduled working period for transmission of all its conflicting nodes (i.e., u's forbidden working periods) stored in $FS(u)$ (lines 12-14).

Algorithm 3: Fast Data Aggregation Scheduling (FDAS)

Initialization: $t_{sch}(u) = 0, t_{min}(u) = 0,$
$FS(u) = \emptyset, n_{uch}(u) = n_{ch}(u)\ \forall u \in V \setminus \{s\}$

```
1: for each unscheduled node u do
2:   if (n_uch(u) = 0) and (rank(u) > {max(rank(i)) |
       ∀i ∈ CS(u)}) then
3:     if t_min(u) = 0 then
4:       t_sch(u) = t_min(u) + 1
5:     else
6:       if A(u) < A(p(u)) then
7:         t_sch(u) = t_min(u)
8:       else
9:         t_sch(u) = t_min(u) + 1
10:      end if
11:    end if
12:    while t_sch(u) ∈ FS(u) do
13:      t_sch(u) = t_sch(u) + 1
14:    end while
15:    u sends a MARK message to its two-hop neighbors
16:    for each unscheduled node v receives MARK do
17:      if u ∈ CS(v) then
18:        FS(v) = FS(v) ∪ {t_sch(u)};
           CS(v) = CS(v) \ {u}
19:      end if
20:      if v = p(u) then
21:        n_uch(v) = n_uch(v) − 1
22:        if t_sch(u) > t_min(v) then
23:          t_min(v) = t_sch(u)
24:        end if
25:      end if
26:    end for
27:  end if
28: end for
```

After finishing the scheduling process, u broadcasts a MARK message containing its ID and scheduled working period $t_{sch}(u)$ to its two-hop neighbors. If an unscheduled conflicting node of u receives this message, since u has already scheduled, it removes u from its conflicting set CS and adds $t_{sch}(u)$ to its FS set (lines 17-19). When receiving the message from u, $p(u)$ decreases the number of its unscheduled child nodes, $n_{uch}(p(u))$, by one and updates its minimum valid working period for transmission, $t_{min}(p(u))$, if $t_{sch}(u)$ is greater than the current $t_{min}(p(u))$ (lines 20-24). Finally, when the FDAS algorithm has finished, all nodes in the network (except the sink s) will transmit their data based on their schedules.

2) Discussion of Xu's Distributed Scheduling Algorithm: In [16], Xu *et al.* propose a distributed algorithm called IAS to schedule the data aggregation of nodes in WSNs. Each node u randomly selects an integer number r_u. A ready node u proceeds in its schedule if and only if r_u is smaller than the number of all its unscheduled ready competitor nodes.

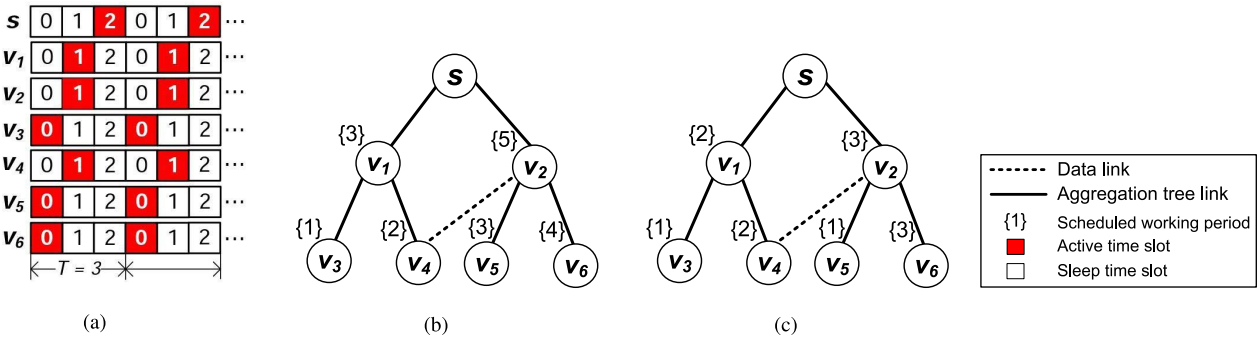


Fig. 5. A comparison example of IAS [16] and FDAS. (a) Duty-cycling schedule coherent with FDAS. (b) Aggregation schedule of IAS ($T = 5$). (c) Aggregation schedule of FDAS ($T = 3$).

The scheduled time slot of a node u, $TST[u]$, is initialized to zero and calculated as follows. When a ready competitor node v of u is scheduled, u will set $TST[u]$ to the larger one between its current $TST[u]$ and $TST[v] + 1$. In FDAS algorithm, the scheduled working period of u, $t_{sch}(u)$, is calculated first based on the schedule of all its child nodes in the data aggregation tree, and the active time slots of u and its parent node $p(u)$. Then to prevent the collisions, this calculated working period of u should be checked as to whether it coincides with any scheduled working period in its set of forbidden working periods. If there is no coincidence, u is scheduled to transmit data in $t_{sch}(u)$. Otherwise, $t_{sch}(u)$ is increased by one and the coincidence is checked again. By this method, the FDAS algorithm allows a node to transmit its recently received data to the parent node in the same working period with its child node without collisions. Thus, the FDAS algorithm reduces significantly the data aggregation delay. Fig. 5 shows a comparison example of IAS and FDAS.

Fig. 5(a) shows the duty-cycling schedule of all nodes in the network. Fig. 5(b) and Fig. 5(c) show the aggregation schedule of nodes when the IAS and FDAS are used, respectively. We assume that $r_{v_1} < r_{v_2} < r_{v_3} < r_{v_4} < r_{v_5} < r_{v_6}$ and all nodes have their sensing data before time slot 0 of working period 1. In IAS, at the beginning, $TST[v_i] = 1$; $i = 1, 2, .., 6$ since nodes can only be scheduled to transmit data from working period 1. Ready nodes are v_3, v_4, v_5, and v_6. v_3 is scheduled first in working period 1 ($TST[v_3] = 1$) and then v_4 is scheduled in working period 2 ($TST[v_4] = 2$). To prevent the collision, v_5 is scheduled in working period $max\{TST[v_5], TST[v_4] + 1\} = max\{1, 3\} = 3$. In contrast to SA, in our proposed algorithm FDAS, first because v_5 is the leaf node in the data aggregation tree, $t_{sch}(v_5) = 1$. This working period is checked with the scheduled working period of v_5's conflicting nodes in $FS(v_5)$. Since $FS(v_5) = \{t_{sch}(v_4)\} = \{2\}$, v_5 can be scheduled to transmit its data to v_2 in working period 1. Also, our proposed scheme allows a node to transmit data in the same working period with its child node if $A(u) < A(p(u))$ and this scheduled working period does not conflict with other transmission schedules (e.g., $t_{sch}(v_1) = t_{sch}(v_4) = 2$ and $t_{sch}(v_2) = t_{sch}(v_6) = 3$). Therefore, the FDAS can reduce significantly the data aggregation delay of IAS. For example, Fig. 5(b) and (c) show that IAS requires 5 working periods instead of 3 working periods of the FDAS for the sink s to receive data from all nodes.

V. Analysis

In this section, first we prove that our proposed scheme, DEDAS-D, constructs a spanning tree rooted at the sink in the first phase and generates a collision-free aggregation schedule in the second phase. Then we present arguments to prove that DEDAS-D is a better approach to solve the MLAS problem in duty-cycled WSNs compared with other existing schemes.

A. Correctness

Theorem 2: DCT algorithm generates a spanning tree rooted at the sink s.

Proof: Let G_{DCT} be the graph whose vertices are the set of sensor nodes and edges are the corresponding links between nodes and their parent nodes determined by the DCT algorithm. We prove that: 1) G_{DCT} does not contain any cycle and has $N - 1$ edges, where N is the number of nodes in the network and 2) G_{DCT} is rooted at s . To support the formal proof of Theorem 2, we state and prove the following lemmas.

Lemma 1: After the start-up stage of the DCT algorithm, each special node has only one parent node in the G_{DCT} and no cycle is created.

Proof: In the start-up stage, all special nodes select their unique neighbor as the parent node. Then these special nodes will change their states from WAIT to COMPLETE. Since nodes in COMPLETE state do not participate in the G_{DCT} construction process any more, they will not create any new edge of the G_{DCT} with other nodes in the network. Thus, after the start-up stage, each special node selects only one neighbor as its parent node and no cycle is generated. ■

Lemma 2: After the main stage of the DCT algorithm, each node in the network (except the sink s and special nodes in the start-up stage) has only one parent node in the G_{DCT} and no cycle is created.

Proof: At the beginning of the main stage, all special nodes in the start-up stage were in COMPLETE state and will not participate in the G_{DCT} construction process in this stage. The sink s initializes this stage by changing its state from WAIT to ACTIVATED (i.e., becoming the first activator). Each activator u will select one node v in WAIT state as the child node. Then v is included in the G_{DCT} and its state will be changed to COMPLETE or ACTIVATED. If $state(v) =$ COMPLETE, v will not participate in the G_{DCT} construction process any more. If $state(v) =$ ACTIVATED,

v becomes a new activator and selects other nodes in WAIT state as its child nodes. Since a node is in WAIT state if and only if it has not been included in the G_{DCT} yet, in both aforementioned cases, node v cannot create any new edge with another node that has belonged to the G_{DCT}. Therefore, after the main stage, each node (except the sink s and special nodes in the start-up stage) has only one parent node in the G_{DCT} and no cycle is generated. ■

Now, we finish the proof of Theorem 2. According to Lemma 1 and Lemma 2, G_{DCT} does not contain any cycle. Each node in the network (except the sink s) selects one node as its parent node in the G_{DCT} and creates only one edge with this parent. Thus, the number of edges of G_{DCT} is $N-1$. The exchange of STUCK messages and rebroadcasting of FIND_CHILD messages in the main stage ensure that for each node in the network, all its neighbors will be included in G_{DCT}. In addition, the construction of G_{DCT} starts from s and ends when s cannot find any neighbor that is not in the G_{DCT}, i.e., all branches of G_{DCT} have already been connected to s. As a result, G_{DCT} is rooted at s. Finally, we conclude that G_{DCT} is a spanning tree rooted at s. ■

Theorem 3: The FDAS algorithm generates a collision-free aggregation schedule.

Proof: We prove that the FDAS algorithm generates a collision-free aggregation schedule by contradiction. Assume that the schedule generated by the FDAS algorithm is not collision-free. This means at least two nodes u and v, whose parent nodes have the same active time slot, are scheduled to send data in the same working period, i.e., $t_{sch}(u) = t_{sch}(v)$. Without loss of generality, we assume that u is scheduled to transmit its data before v in working period $t_{sch}(u)$. For each unscheduled conflicting node of u, the scheduled working period for u's transmission, $t_{sch}(u)$, should be added to its set of forbidden working periods (Algorithm 3, lines 17-19). Since v is a conflicting node of u, $t_{sch}(u) \in FS(v)$. When v is being scheduled, the loop (Algorithm 3, lines 12-14) is terminated if and only if $t_{sch}(v) \neq t_{sch}(i)\ \forall t_{sch}(i) \in FS(v)$. Therefore, $t_{sch}(v) \neq t_{sch}(u)$. ■

B. Discussion of Existing Schemes Solving the MLAS Problem in Duty-cycled WSNs

DEDAS-D is a better solution to solve the MLAS problem in duty-cycled WSNs compared with previous schemes because of the following underlying reasons.

- **Considering the impact of sleep latency on data aggregation delay.** Previous schemes do not consider the duty cycle [16], [24], [25] or only consider the active time slot of a receiver but do not consider sleep latency between sender and receiver when scheduling a node [17]. However, this sleep latency can affect the data aggregation delay [15]. Because DEDAS-D considers sleep latency between nodes when building the data aggregation tree in the first phase as well as scheduling a node in the second phase, the data aggregation delay is decreased significantly.
- **Considering the impact of high degree nodes in the data aggregation tree on data aggregation delay.** Many existing approaches solving the MLAS problem [16], [17], [22]–[24] construct the data aggregation tree based on finding MIS or CDS. However, these data aggregation trees also have a high-degree bottleneck problem, i.e., many high degree nodes exist in the tree and thus reduce the number of concurrent transmissions. As shown in Theorem 1, the data aggregation delay is increased if those structures are used. Thanks to the degree-constrained aggregation tree, we can significantly reduce the impact of high degree nodes on the data aggregation delay.
- **Proposing an advanced method to generate a collision-free schedule.** To prevent the collisions, in SA [17], the working periods of all nodes in the i-th layer of the data aggregation tree are delayed by the latest assigned working period of nodes in deeper layers. This leads to unnecessary waiting for transmission of nodes in the same level and high data aggregation delay. Our proposed algorithm FDAS can generate a pipelined schedule to allow more nodes to transmit concurrently by considering the maximum scheduled working period of child nodes and active time slots of both sender and receiver. Therefore, the FDAS diminishes the non-essential waiting of SA and reduces the data aggregation delay. Through the conflicting set and the forbidden set information of each node, as proved in Theorem 3, a collision-free aggregation schedule is generated by FDAS algorithm. Moreover, in comparison with a best-known distributed scheduling algorithm, IAS [16], as shown in Section IV, FDAS also outperforms IAS in terms of data aggregation delay.
- **Solving the MLAS problem in duty-cycled WSNs by a distributed approach.** Our previously proposed centralized scheme DEDAS-B [18] overcomes the shortcomings of SA and achieves the best delay performance among existing schemes. Nevertheless, DEDAS-B is a centralized approach. To the best of our knowledge, DEDAS-D is the first distributed scheme solving the MLAS problem in duty-cycled WSNs. Therefore, it is more suitable for practical implementation than other centralized algorithms [17], [18]. In the next section, we show that the delay performance of DEDAS-D approximates to that of DEDAS-B. Thus, we can conclude that DEDAS-D is a better approach to solve the MLAS problem in duty-cycled WSNs compared with previous approaches.

VI. Performance Evaluation

We use a simulator built in MATLAB [36] to evaluate the data aggregation delay of the proposed scheme DEDAS-D under different values of degree threshold α, number of sensor nodes, network densities, and duty cycles. First, we investigate the impact of degree threshold α on the delay performance of DEDAS-D and select values of α that minimize the data aggregation delay of proposed scheme for our next experiments. Based on selected values of α, we compare the data aggregation delay of DEDAS-D with that of IAS [16], SA [17], and DEDAS-B [18].

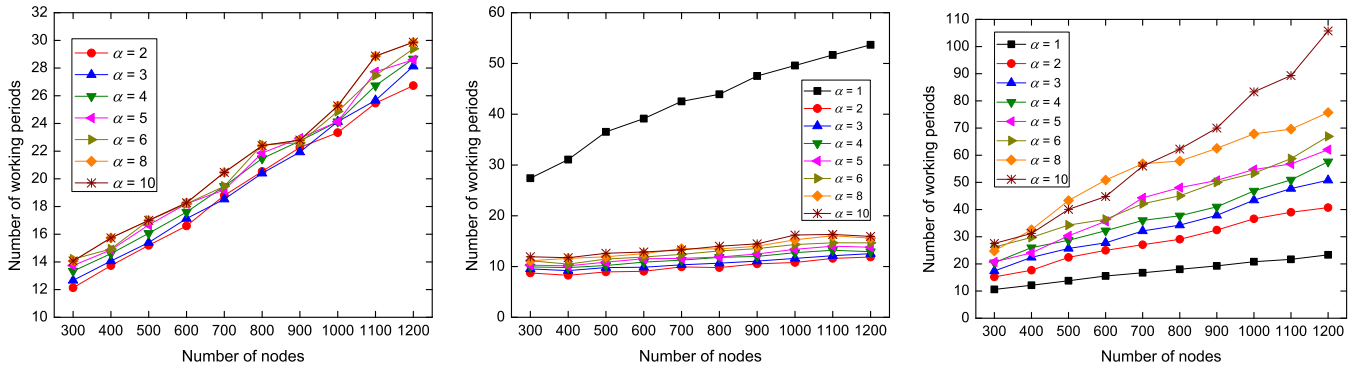


Fig. 6. The impact of the number of nodes on data aggregation delay of DEDAS-D with various α. (a) $L = 200m$, $T = 10$. (b) $L = 200m$, $T = 100$. (c) $L = 50m$, $T = 100$.

We randomly deploy N sensor nodes in a square region with side length L. All sensor nodes have the same transmission range $R = 30$m. The sink s is located in the bottom left corner of the field. Each sensor node randomly selects a time slot in $[0, T-1]$ as its active time slot in each working period. We use the data aggregation delay as a metric to evaluate the performance of schemes. The data aggregation delay is defined as the required number of working periods for the sink to receive data from all other nodes in the network. We carry out experiments in three different scenarios: 1) varying the number of nodes N, 2) varying the network side length L, and 3) varying the duty cycle by varying the number of time slots T in a working period. For each set of parameter configurations, the presented results are the average of 15 runs on random topologies.

A. Impact of Degree Threshold α on Data Aggregation Delay of DEDAS-D

We first evaluate the data aggregation delay of our proposed scheme DEDAS-D with different values of degree threshold α under various network settings. The DEDAS-D scheme with α is denoted as DEDAS-D$_\alpha$.

1) Varying Number of Nodes: In this test, the number of nodes N is varied from 300 to 1200. The delay results of three network settings with different network side lengths and duty cycles are depicted in Fig. 6(a), (b), and (c). Since DEDAS-D$_1$ has 77.82% delay higher than DEDAS-D$_{10}$ on average, we omit the results of DEDAS-D$_1$ in Fig. 6(a) for a better visualization. Fig. 6 shows that the data aggregation delay of DEDAS-D$_\alpha$; $\alpha = 1, 2, ..., 10$ increases as the number of nodes increases due to the overall increase of traffic needed to be transmitted. Additionally, the collision probability also grows when the number of nodes increases causing an increment of data aggregation delay.

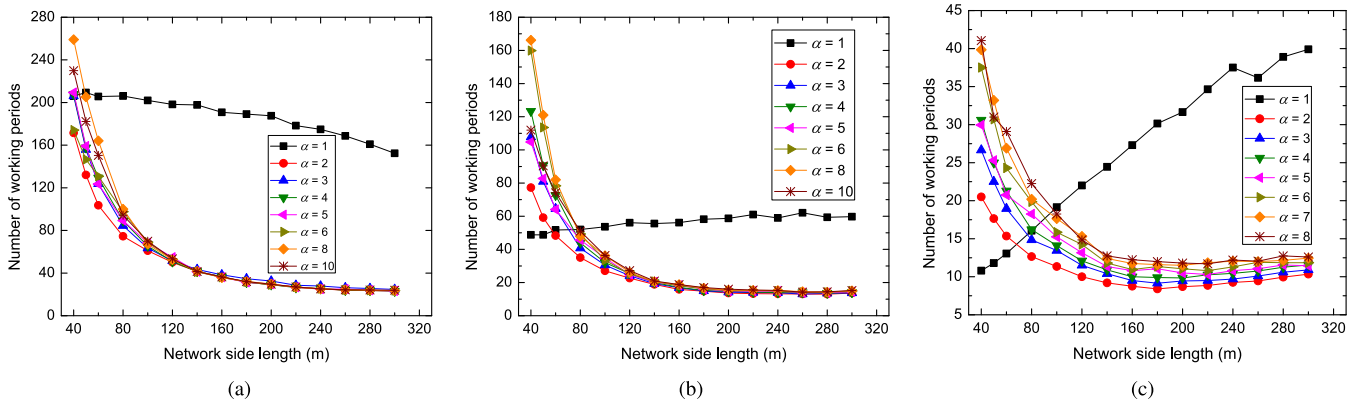
From Fig. 6(a) and (b), we observe that DEDAS-D$_2$ outperforms DEDAS-D with other values of α in terms of data aggregation delay. However, the effectiveness of DEDAS-D$_1$ increases significantly in high-density and low duty-cycled networks (i.e., small L and large T). As shown in Fig. 6(c), DEDAS-D$_1$ improves the performance of DEDAS-D$_2$ by 38.51%.

TABLE I
SIDE LENGTH VS. AVERAGE NUMBER OF NEIGHBORS

Network side length (m)	Average number of neighbors
40	319.23
50	247.91
60	191.56
80	123.63
100	85.60
120	62.24
140	47.96
160	37.82
180	30.28
200	24.82
220	20.58
240	17.36
260	14.97
280	13.10
300	11.54

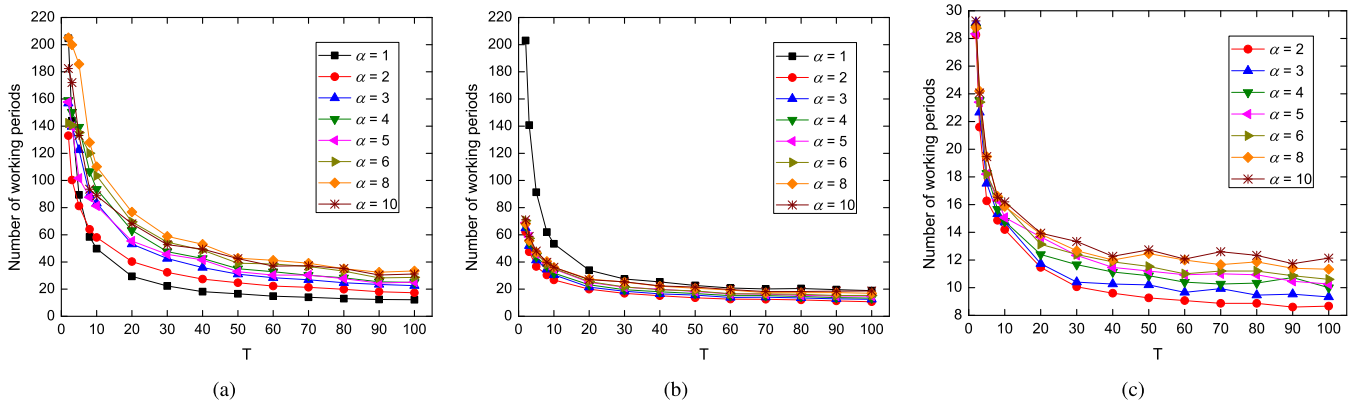
2) Varying Network Side Length: We next compare the data aggregation delay achieved by DEDAS-D with various α in different network side lengths. The number of nodes N is fixed at 400 while the network side length L is varied from 40 to 300m. Table I shows the corresponding average number of neighbors of a sensor node with different network side lengths. The average number of neighbors reflects the network density. The higher the average number of neighbors a sensor node has, the higher the network density. Therefore, the network density is inversely proportional to the network side length L. For example, when $L = 40$m, one node is within the transmission range of 320 other nodes (about 80% of the total number of nodes in the network) whereas when $L = 300$m, the number of average neighbors is reduced to 12 nodes (only about 3% of the total number of nodes in the network).

The results of scenarios where the duty cycle is set to 50%, 10%, and 1% corresponding to $T = 2, 10$, and 100 are presented in Fig. 7(a), (b), and (c), respectively. Fig. 7(a) shows that the data aggregation delay of DEDAS-D$_\alpha$; $\alpha = 1, 2, ..., 10$ increases as the sensor deployment gets denser. The reason for this trend is many nodes are each other's neighbors in dense networks. Thus, the higher number of working periods for data aggregation is required to prevent the collisions. In high duty-cycled networks (i.e., small T),



(a) (b) (c)

Fig. 7. The impact of the network side length on data aggregation delay of DEDAS-D with various α. (a) $N = 400$, $T = 2$. (b) $N = 400$, $T = 10$. (c) $N = 400$, $T = 100$.



(a) (b) (c)

Fig. 8. The impact of the duty cycle on data aggregation delay of DEDAS-D with various α. (a) $N = 400$, $L = 50m$. (b) $N = 400$, $L = 100m$. (c) $N = 400$, $L = 200m$.

DEDAS-D_2 achieves the lowest data aggregation delay in all network densities (as shown in Fig. 7(a)).

From Fig. 7(a), (b), and (c), we can observe that the effectiveness of DEDAS-D_1 decreases as T decreases. The reason is that the probability for a node to transmit data in the same working period with its received working period is reduced as T decreases. Because DEDAS-D_1 generates a path structure as the aggregation tree, when T gets smaller, most of nodes should wait until the next working period to send its received data to the parent; thus, its data aggregation delay increases.

For the same T value, the performance of DEDAS-D_1 decreases significantly as the network side length L increases. For example, as shown in Fig. 7(b), when L is in the range of 40 to 60m, DEDAS-D_1 outperforms other compared schemes in terms of data aggregation delay. However, once L increases beyond 60m, the data aggregation delay of DEDAS-D_1 exceeds that of DEDAS-D_2. In particular, when L is greater than 80m, DEDAS-D_1 has the highest data aggregation delay compared with other schemes. The reason for the performance degradation of DEDAS-D_1 when L increases is that in sparse networks, node has less opportunities to select a parent in the DCT among its neighbors that allows both child node and parent node to transmit in the same working period. The results shown in Fig. 7(a), (b), and (c) indicate that network density and duty cycle are two dominant influential factors in the delay performance of DEDAS-D_1.

3) Varying Duty Cycle: Fig. 8 illustrates the variation of data aggregation delay with duty cycle for different values of α. We fix the number of sensor nodes at 400 and vary the duty cycle from 50% to 1% corresponding to T from 2 to 100. Again, for the similar reason mentioned in part 1, we remove the results of DEDAS-D_1 from Fig. 8(c). It can be seen from Fig. 8 that when T increases, the required number of working periods for data aggregation of all schemes decreases. This happens because the increase of T leads to an increase in opportunities for nodes to transmit concurrently without collisions. Therefore, data aggregation delay is reduced.

Furthermore, the results shown in Fig. 8 also corroborates our observation in previous parts that DEDAS-D_1 achieves the best delay performance only in high-density and low duty-cycled networks. Otherwise, in sparser scenarios or higher duty-cycled networks, the performance of DEDAS-D_2 is much better than that of DEDAS-D with other α values.

From the above evaluations, we can conclude that a suitable value of α ($\alpha = 1$ or $\alpha = 2$) can be selected based on the network density and duty cycle to minimize the data aggregation delay. For the next experiments, we compare the delay performance of DEDAS-D_1 and DEDAS-D_2 with those achieved by other related schemes.

B. Delay Comparison of DEDAS-D and Other Schemes

In this section, we compare the delay performance of our proposed scheme DEDAS-D with that of IAS, SA, and

TABLE II
LIST OF SCHEMES IMPLEMENTED IN THE PERFORMANCE EVALUATION STUDY

Scheme	Centralized vs. distributed	Tree construction algorithm	Scheduling algorithm
SA	Centralized	Maximal independent set based	Minimal covering schedule based
DEDAS-B	Centralized	Balancing SPT	DEDAS
DCT_0 + IAS	Distributed	Connected dominating set based	IAS
DCT_0 + FDAS	Distributed	Connected dominating set based	Proposed FDAS
DCT_1 + IAS	Distributed	Proposed DCT with $\alpha = 1$	IAS
DCT_2 + IAS	Distributed	Proposed DCT with $\alpha = 2$	IAS
DCT_1 + FDAS (DEDAS-D_1)	Distributed	Proposed DCT with $\alpha = 1$	Proposed FDAS
DCT_2 + FDAS (DEDAS-D_2)	Distributed	Proposed DCT with $\alpha = 2$	Proposed FDAS

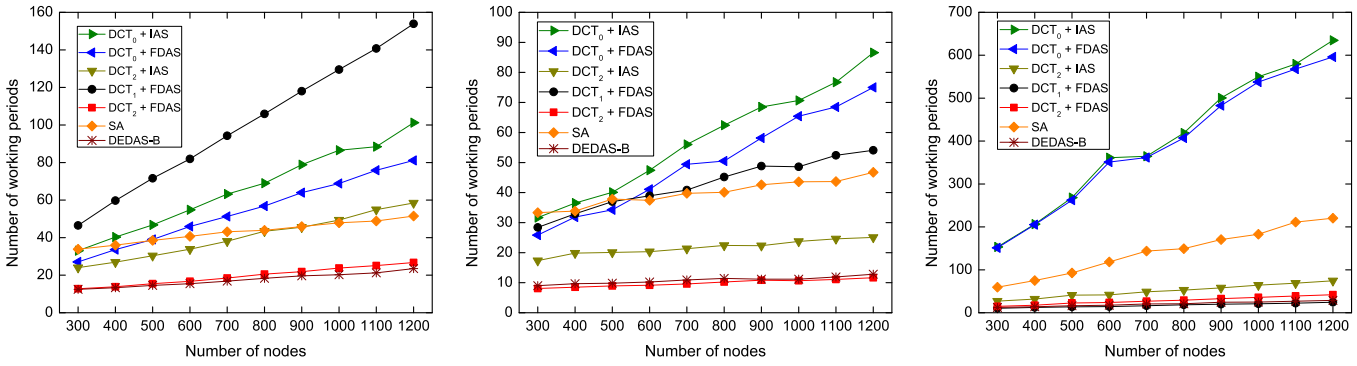


Fig. 9. Data aggregation delay comparison of schemes with different number of nodes. (a) $L = .200m$, $T = 10$. (b) $L = 200m$, $T = 100$. (c) $L = 50m$, $T = 100$.

DEDAS-B under different network settings. Because IAS does not consider a duty-cycle, we make a modification on this scheme to adapt it to a duty-cycled environment: given the output schedule $< M, v >$ of a node u which means u sends its data to v in time slot M, we make u send data to v in $A(v)$ of M-th working period.

In addition, to evaluate in detail the performances of our two novel algorithms in DEDAS-D (i.e., the DCT algorithm and the FDAS algorithm), we combine each proposed algorithm with the tree construction algorithm or scheduling algorithm proposed in [16] to generate three new schemes named DCT_0 + FDAS, DCT_1 + IAS, and DCT_2 + IAS, where DCT_0 denotes the connected dominating set based tree construction algorithm proposed in [16]. Table II shows the list of schemes implemented in our performance evaluation study. The original scheme in [16] is denoted by DCT_0 + IAS and our proposed schemes DEDAS-D_1 and DEDAS-D_2 are highlighted in grey.

1) Varying Number of Nodes: Fig. 9 demonstrates the data aggregation delay of schemes when the number of nodes N is varied from 300 to 1200. We consider three topologies with different network side lengths and duty cycles. It should be noted that the scheme DCT_1 + IAS always has the worst delay performance compared with other schemes (e.g., the data aggregation delay of DCT_1 + IAS is 6.74, 11.65, and 1.86 times higher than that of DCT_1 + FDAS in Fig. 9(a) and DCT_0 + IAS in Fig. 9(b) and (c), respectively). Therefore, we do not show the performance of DCT_1 + IAS in Fig. 9 for a better visualization.

DEDAS-D_2 reduces the data aggregation delay of the first centralized algorithm solving the MLAS problem in duty-cycled WSNs, SA, by 55.21% and 75.26% as shown in Fig. 9(a) and (b), respectively. The performance of DEDAS-D_1 increases as the network side length decreases. For example, in Fig. 9(b), when the network side length is 200m, the data aggregation delay of DEDAS-D_1 is 4.88% and 76.51% higher than that of SA and DEDAS-D_2. Fig. 9(c) indicates that when the network side length is reduced by a factor of 4, DEDAS-D_1 can achieve 44.50% and 27.73% less data aggregation delay compared with that of SA and DEDAS-D_2.

From Fig. 9, we observe that our previous centralized scheme DEDAS-B has the lowest data aggregation delay among compared schemes. However, the performance gap between DEDAS-B and our proposed distributed scheme DEDAS-D (including DEDAS-D_1 in high-density and low duty-cycled networks, and DEDAS-D_2 in the remaining scenarios) are negligible. DEDAS-D achieves superior performance compared with SA and the distributed scheme in [16] because it reduces the effect of high degree nodes by constructing the DCT and allows more nodes to transmit concurrently in the same working period by carrying out the FDAS algorithm. Moreover, since both DCT_0 + FDAS and DCT_2 + IAS schemes outperform DCT_0 + IAS, we can conclude that both the DCT algorithm and the FDAS algorithm can improve the delay performance of corresponding algorithms proposed in [16].

2) Varying Network Side Length: In this experiment, we fix the number of sensor nodes N at 400 and vary the network side length L from 40 to 300m. By increasing L, the average number of neighbors is reduced and thus the network density decreases. The results when the duty cycle is set to 50%, 10%, and 1% corresponding to $T = 2, 10$, and 100 are shown in Fig. 10(a), (b), and (c), respectively.

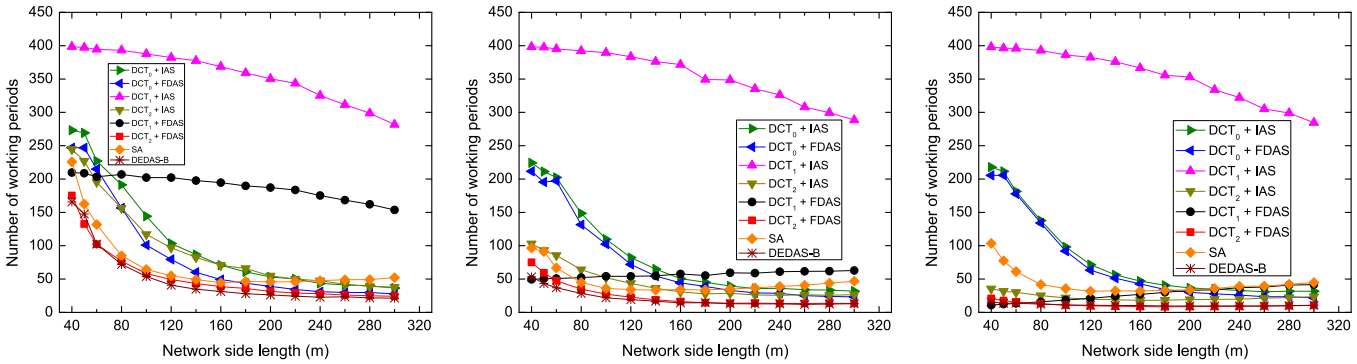


Fig. 10. Data aggregation delay comparison of schemes with different network side lengths. (a) $N = 400$, $T = 2$. (b) $N = 400$, $T = 10$. (c) $N = 400$, $T = 100$.

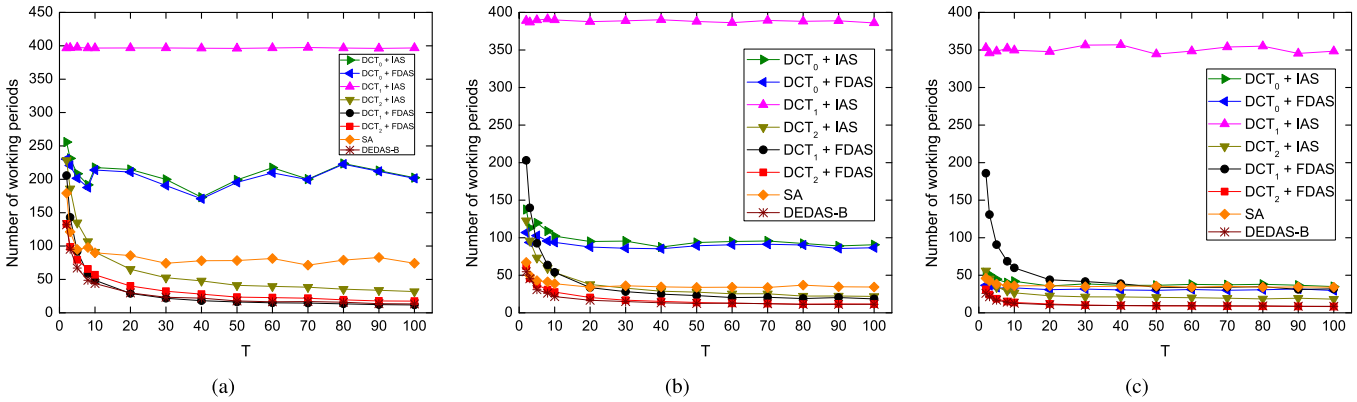


Fig. 11. Data aggregation delay comparison of schemes with different duty cycles. (a) $N = 400$, $L = 50m$. (b) $N = 400$, $L = 100m$. (c) $N = 400$, $L = 200m$

It can be seen that as the duty cycle decreases (i.e., T increases), the improvement of DEDAS-D_1 increases significantly. For the same duty cycle, the effectiveness of DEDAS-D_1 increases proportionally to the increase of network density. The results in Fig. 10(a) reveals that in high duty-cycled networks, the delay of DEDAS-D_1 is much higher than that of DEDAS-D_2 in all network densities. In low duty-cycled networks (e.g., $T = 10$ or $T = 100$), when the network density is low (e.g., L is greater than 60m), DEDAS-D_2 still outperforms DEDAS-D_1.

However, when L is smaller than 60m, DEDAS-D_1 achieves lower data aggregation delay compared with DEDAS-D_2 as illustrated in Fig. 10(b) and (c). The reason is that along with the increase of network density and T, the probability for a node to select a parent which allows concurrent transmissions increases. As a result, in dense and low duty-cycled networks, the DCT_1 is a better structure than other data aggregation trees to maximize the number of parallel transmissions and reduces the required number of working periods for aggregating all data in the network. Nevertheless, when the network gets sparser and T gets smaller, the DCT_1 is not suitable for such scenarios since it decreases the number of nodes can transmit in the same working period and thus incurs higher data aggregation delay.

Fig. 10 shows that our proposed scheme DEDAS-D including DEDAS-D_1 and DEDAS-D_2 improves the delay performance of DCT_0 + IAS and SA by 65.04% and 50.95% on average, respectively. Furthermore, the performance of DEDAS-D is close to that of DEDAS-B with all network side lengths.

3) Varying Duty Cycle: Fig. 11(a), (b) and (c) illustrate the required number of working periods for data aggregation of compared schemes when T is varied from 2 to 100 and L is fixed at 50, 100, and 200m. The simulation results show a similar trend inferred from Fig. 10, i.e, the delay performance of DEDAS-D_1 surpasses that of SA and DEDAS-D_2 only in high-density and low duty-cycled networks. For instance, Fig. 11(a) shows that when L is 50m, DEDAS-D_1 achieves a reduction of 14.23% and 54.49% on average in the required number of working periods as compared with DEDAS-D_2 and SA. Moreover, the higher the value of T is, the better the delay performance of DEDAS-D_1 is. In contrast, from Fig. 11(b) and (c), when L is 100 or 200m, we can see that DEDAS-D_2 has a smaller delay than DEDAS-D_1 and SA under all duty cycles. It is worth noting that in all cases, our proposed scheme DEDAS-D ($\alpha = 1$ or $\alpha = 2$) achieves an asymptotic performance compared with our previous centralized scheme DEDAS-B which has the best delay performance. On average, DEDAS-D improves the performance of DCT_0 + IAS and SA by 75.80% and 56.58%, respectively.

VII. Conclusion

In this paper, we propose a distributed delay-efficient scheme named DEDAS-D to solve the MLAS problem in

duty-cycled WSNs for the first time. DEDAS-D consists of two new algorithms in data aggregation tree construction and scheduling. We carry out extensive simulations to evaluate the performance of our proposed scheme in terms of data aggregation delay. The simulation results show that both proposed algorithms outperform corresponding existing distributed algorithms solving the MLAS problem. Comparing with the first centralized scheme solving the MLAS problem in duty-cycled WSNs, the data aggregation delay of DEDAS-D is reduced by 50.95% and 56.58% while varying network side length and duty cycle, respectively. Furthermore, DEDAS-D has a comparable delay performance as our previously proposed centralized scheme DEDAS-B. In our future work, we will determine the appropriate degree threshold α and duty cycle to minimize the data aggregation delay. Additionally, we have a plan to study the MLAS problem in duty-cycled WSNs under the physical interference (real test bed) model.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[2] D. Puccinelli and M. Haenggi, "Wireless sensor networks: Applications and challenges of ubiquitous sensing," *IEEE Circuits Syst. Mag.*, vol. 5, no. 3, pp. 19–31, Sep. 2005.

[3] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: A tiny aggregation service for ad-hoc sensor networks," *ACM SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 131–146, Dec. 2002.

[4] B. Kang, N. Kwon, and H. Choo, "Developing route optimization-based PMIPv6 testbed for reliable packet transmission," *IEEE Access*, vol. 4, pp. 1039–1049, 2016.

[5] O. D. Incel, A. Ghosh, and B. Krishnamachari, "Scheduling algorithms for tree-based data collection in wireless sensor networks," in *Theoretical Aspects of Distributed Computing in Sensor Networks*. Berlin, Germany: Springer, 2011, pp. 407–445.

[6] B. Kang and H. Choo, "An SDN-enhanced load-balancing technique in the cloud system," *J. Supercomputing*, vol. 72, no. 1, pp. 1–24, 2016.

[7] B. Kang and H. Choo, "A cluster-based decentralized job dispatching for the large-scale cloud," *EURASIP J. Wireless Commun. Netw.*, vol. 2016, no. 1, pp. 1–8, 2016.

[8] X. Fafoutis, A. D. Mauro, M. D. Vithanage, and N. Dragoni, "Receiver-initiated medium access control protocols for wireless sensor networks," *Comput. Netw.*, vol. 76, pp. 55–74, Jan. 2015.

[9] N. Kumar, S. Misra, and M. S. Obaidat, "Collaborative learning automata-based routing for rescue operations in dense urban regions using vehicular sensor networks," *IEEE Syst. J.*, vol. 9, no. 3, pp. 1081–1090, Sep. 2015.

[10] B. Kang, S. Myoung, and H. Choo, "Distributed degree-based link scheduling for collision avoidance in wireless sensor networks," *IEEE Access*, vol. 4, pp. 7452–7468, 2016.

[11] S.-G. Jung, B. Kang, S. Yeoum, and H. Choo, "Trail-using ant behavior based energy-efficient routing protocol in wireless sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 2016, no. 1, pp. 1–8.

[12] M. Aldeer, R. Howard, and A. Al-Hilli, "Minimizing energy consumption in transmit-only sensor networks via optimal placement of the cluster heads," in *Proc. 8th Wireless of the Students, by the Students, and for the Students Workshop*, 2016, pp. 36–38.

[13] A. Razaque and K. M. Elleithy, "Low duty cycle, energy-efficient and mobility-based boarder Node—MAC hybrid protocol for wireless sensor networks," *J. Signal Process. Syst.*, vol. 81, no. 2, pp. 265–284, 2015.

[14] G. Han, Y. Dong, H. Guo, L. Shu, and D. Wu, "Cross-layer optimized routing in wireless sensor networks with duty cycle and energy harvesting," *Wireless Commun. Mobile Comput.*, vol. 15, no. 16, pp. 1957–1981, Nov. 2015.

[15] Y. Gu and T. He, "Dynamic switching-based data forwarding for low-duty-cycle wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 12, pp. 1741–1754, Dec. 2011.

[16] X. Xu, X. Y. Li, X. Mao, S. Tang, and S. Wang, "A delay-efficient algorithm for data aggregation in multihop wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 1, pp. 163–175, Jan. 2011.

[17] B. Yu and J.-Z. Li, "Minimum-time aggregation scheduling in duty-cycled wireless sensor networks," *J. Comput. Sci. Technol.*, vol. 26, no. 6, pp. 962–970, 2011.

[18] N. P. K. Ha, V. Zalyubovskiy, and H. Choo, "Delay-efficient data aggregation scheduling in duty-cycled wireless sensor networks," in *Proc. ACM Res. Appl. Comput. Symp.*, 2012, pp. 203–208.

[19] M. Rezvani, A. Ignjatovic, E. Bertino, and S. Jha, "Secure data aggregation technique for wireless sensor networks in the presence of collusion attacks," *IEEE Trans. Depend. Sec. Comput.*, vol. 12, no. 1, pp. 98–110, Jan. 2015.

[20] X.-Y. Liu *et al.*, "CDC: Compressive data collection for wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 8, pp. 2188–2197, Aug. 2014.

[21] X. Chen, X. Hu, and J. Zhu, "Minimum data aggregation time problem in wireless sensor networks," in *Proc. Int. Conf. Mobile Ad-Hoc Sensor Netw.*, 2005, pp. 133–142.

[22] S. C.-H. Huang, P.-J. Wan, C. T. Vu, Y. Li, and F. Yao, "Nearly constant approximation for data aggregation scheduling in wireless sensor networks," in *Proc. 26th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, May 2007, pp. 366–372.

[23] P.-J. Wan, S. C.-H. Huang, L. Wang, Z. Wan, and X. Jia, "Minimum-latency aggregation scheduling in multihop wireless networks," in *Proc. 10th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2009, pp. 185–194.

[24] B. Yu, J. Li, and Y. Li, "Distributed data aggregation scheduling in wireless sensor networks," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 2159–2167.

[25] Y. Li, L. Guo, and S. K. Prasad, "An energy-efficient distributed algorithm for minimum-latency aggregation scheduling in wireless sensor networks," in *Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2010, pp. 827–836.

[26] Y. Gu, T. He, M. Lin, and J. Xu, "Spatiotemporal delay control for low-duty-cycle sensor networks," in *Proc. 30th IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2009, pp. 127–137.

[27] S. Guo, S. M. Kim, T. Zhu, Y. Gu, and T. He, "Correlated flooding in low-duty-cycle wireless sensor networks," in *Proc. 19th IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2011, pp. 383–392.

[28] X. Jiao, W. Lou, J. Ma, J. Cao, X. Wang, and X. Zhou, "Minimum latency broadcast scheduling in duty-cycled multihop wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 1, pp. 110–117, Jan. 2012.

[29] F. Wang and J. Liu, "On reliable broadcast in low duty-cycle wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 5, pp. 767–779, May 2012.

[30] J. Hong, J. Cao, W. Li, S. Lu, and D. Chen, "Minimum-transmission broadcast in uncoordinated duty-cycled wireless ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 1, pp. 307–318, Jan. 2010.

[31] S. Lai and B. Ravindran, "On multihop broadcast over adaptively duty-cycled wireless sensor networks," in *Proc. Int. Conf. Distrib. Comput. Sensor Syst.*, 2010, pp. 158–171.

[32] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," *Wireless Netw.*, vol. 11, no. 4, pp. 471–487, Jul. 2005.

[33] S. Ramanathan and E. L. Lloyd, "Scheduling algorithms for multihop radio networks," *IEEE/ACM Trans. Netw.*, vol. 1, no. 2, pp. 166–177, Apr. 1993.

[34] M. R. Gary and D. S. Johnson, "Computers and intractability: A guide to the theory of np-completeness," *SIAM Rev.*, vol. 24, no. 1, pp. 90–91, 1979.

[35] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*. Philadelphia, PA, USA: SIAM, 2000.

[36] M. Grant, S. Boyd, and Y. Ye, "CVX: Matlab software for disciplined convex programming," CVX Res., Inc., Austin, TX, USA, Tech. Rep. 1116, 2008.

Byungseok Kang received the B.S. degree in computer engineering from Sejong University, South Korea, in 2006, the M.S. degree in electrical and electronics engineering from Korea University, South Korea, in 2008, and the Ph.D. degree in electronics and computer science from the University of Southampton, U.K., in 2015. He is currently a Research Professor with Sungkyunkwan University since 2015. His research interests include wired/wireless networking, sensor networking, mobile computing, network security protocols, and simulations/numerical analysis.

Phan Khanh Ha Nguyen received the B.S. degree in electronics and telecommunications engineering from the Hanoi University of Science and Technology, Vietnam, in 2010, and the M.S. degree from the Department of Electrical and Computer Engineering, College of Information and Communication Engineering, Sungkyunkwan University, South Korea, in 2013. His research interests include delay and energy efficient scheduling algorithms in wireless sensor networks and wireless communication protocols.

Vyacheslav Zalyubovskiy received the M.S. degree in mathematics from Novosibirsk State University, Russia, and the Ph.D. degree in mathematics from the Sobolev Institute of Mathematics of the Siberian Branch, Russian Academy of Sciences. Since 1984, he has been a Scientific Researcher with the Sobolev Institute of Mathematics. In 2007, he joined the College of Information and Communication Engineering, Sungkyunkwan University, as a Research Professor. His recent research focuses on the design and analysis of exact and approximation algorithms for combinatorial optimization problems and their applications, particularly in the areas of networking, planning, and scheduling.

Hyunseung Choo (M'17) received the B.S. degree in mathematics from Sungkyunkwan University, South Korea, in 1988, the M.S. degree in computer science from the University of Texas at Dallas, USA, in 1990, and the Ph.D. degree in computer science from the University of Texas at Arlington, USA, in 1996. From 1997 to 1998, he was a Patent Examiner with Korean Industrial Property Office. In 1998, he joined the College of Information and Communication Engineering, Sungkyunkwan University, and is an Associate Professor and the Director of the Convergence Research Institute. Since 2005, he has been the Director of the Intelligent HCI Convergence Research Center (eight-year research program) supported by the Ministry of Knowledge Economy (Korea) under the Information Technology Research Center support program supervised by the Institute of Information Technology Assessment. He has published over 200 papers in international journals and refereed conferences. His research interests include wired/wireless/optical embedded networking, mobile computing, and grid computing. He is a member of ACM. He is the Vice President of the Korean Society for Internet Information (KSII). He has been the Editor-in-Chief of the Journal of KSII for three years and journal editors of the *Journal of Communications and Networks*, the *ACM Transactions on Internet Technology*, the *International Journal of Mobile Communication*, *Transactions on Computational Science Journal* (Springer-Verlag), and the Editor of the *KSII Transactions on Internet and Information Systems* since 2006.