Delay-Efficient Energy-Minimized Data Collection With Dynamic Traffic in WSNs

Byungseok Kang, Phuc Nguyen, and Hyunseung Choo, *Member, IEEE*

***Abstract*— Data collection is one of the most important applications in wireless sensor networks where sensed data are gathered from sensor nodes to the base station. Reporting redundant data leads to the wastage of time and energy, the sensors therefore report only meaningful information to the base station, which are independent and distinct with the lasted ones. This assumption leads to unpredictable changes of data traffic over different sampling intervals in data collection process. In this paper, we first formulate the tight constraints of the problem and then propose a delay-efficient traffic adaptive (DETA) scheme for collecting data from sensor nodes with minimum energy consumption. The DETA scheme minimizes data collection delay by constructing delay-efficient, collision-free schedule, and by using a special mechanism to enable every node to self-adapt with the changes of data traffic. We also conducted simulations to evaluate the performance of the proposed scheme, and the results shown that the proposed scheme significantly decreases data collection delay and energy consumption compared with the existing schemes.**

***Index Terms*— Wireless sensor networks, data collection, dynamic traffic, scheduling.**

I. Introduction

In DATA collection wireless sensor networks (WSNs), energy efficiency and delay efficiency are two major considerations which have attracted great attention in recent years. To obtain these objectives, constructing efficient scheduling strategy is one of common ways. An efficient schedule would exploit time spaces and therefore reduces latency and energy consumed by data collection. Most of recent research efforts have investigated for full traffic data collection WSNs where the set of reporting nodes remains unchanged over different sampling intervals [32], [33]. In contrast, real monitoring applications are usually operating under the assumption of dynamic traffic scenario where a sensor node only reports its sensed data only if a set of predefined thresholds have been reached. For example:

- To reduce the energy and time of storage and computation caused by data redundancy, the sensor only updates new sensing values with the base station if the new sensed value is different considerably with the previous ones [1], [2].
- To meet the nature constrain of monitoring applications, a sensor only reports its data if the sensed value satisfied the predefined conditions. For example, the sensors in volcano [3] and structural damage [29] monitoring applications only report to the base station if seismic or damage signals have been detected.

In such situations, the scheduling algorithms constructed for full traffic lead to high latency, low throughput, especially when data traffic on the network is progressively light. Obviously, one straightforward approach is to identify the data distribution and then re-construct a new schedule if the data traffic has any change. However, in most cases, it is an inefficient solution due to the costs of extra energies and latency overheads. To solve this problem, Zhao and Tang [4] propose a scheduling algorithm, called Traffic Patterns Oblivious (TPO), which effectively deals with dynamic traffic data collection problem. However, in TPO, the nodes closer to the sink cannot report their data earlier than its descendant even there is no violation of the interference. Therefore, unnecessary delays still occur during data collection process according to the drawback of the scheduling algorithm.

In this paper, we introduce a scheme for delay-efficiency and energy-efficiency in dynamic traffic data collection WSNs. The following lists our key findings and contributions:

- *An efficient data collection schedule*. We propose a scheduling algorithm to assign time slots to sensor nodes respecting interference constraints to minimize data collection delay and energy consumption. This algorithm enables sensor nodes and their descendants to send data simultaneously if the collision-free conditions are satisfied.
- *Traffic adaptive mechanism*. We design an adaptive mechanism to allow schedule algorithm dealing with dynamic traffic. This mechanism enables sensor nodes to reduce their idle listening by identifying the end of transmission performed by the sending node and then stop listening without any risk of missing data. More importantly, the schedule results from the algorithm also make sensor nodes effectively deal with dynamic traffic scenario.

The remainder of this paper is organized as follows. In Section 2, we briefly review the related work. The problem formulation is provided in Section 3. In Section 4, we describe the proposed scheme. The performance evaluation is shown in Section 5. Finally, we conclude the paper in the last section.

Manuscript received December 14, 2017; accepted December 23, 2017. Date of publication January 1, 2018; date of current version March 9, 2018. This work was supported in part by MOE and MSIP, South Korean Government, through the G-ITRC support program under Grant IITP-2017-2015-0-00742, in part by ICT Research and Development program under Grant B0101-15-1366, and in part by the Development of Core Technology for Autonomous Network Control and Management through IITP and PRCP under Grant NRF-2010-0020210. The associate editor coordinating the review of this paper and approving it for publication was Prof. Okyay Kaynak. *(Corresponding author: Hyunseung Choo.)*

B. Kang is with the Department of Data Science, Sejong University, Seoul 05006, South Korea (e-mail: bskang@sejong.ac.kr).

P. Nguyen and H. Choo are with the College of Information and Communication Engineering, Sungkyunkwan University, Suwon 16419, South Korea (e-mail: nvp2803@skku.edu; choo@skku.edu).

Digital Object Identifier 10.1109/JSEN.2017.2788409

II. Related Work

For data collection WSNs, a routing infrastructure has to be constructed to transport data from sensor nodes to the base station due to a limited transmission range. A common practice is to organize the sensor nodes into a tree structure root at the base station [16], [17]. We call it as tree-based data collection problem in WSNs. This problem consists of two types: *non-aggregate data collection* and *aggregate data collection*. In non-aggregate data collection, the base station collects all data packets from sensor nodes individually without any aggregation process. An internal node in the data collection tree needs more transmission slots than any of its descendants. Because it has not only to send its own data but also to relay all the data received from its descendants to its parent [4], [7]–[12]. In aggregate data collection, the sensor node compresses the data from all its children and its own data and then sends the aggregated data to its parent [18]–[23], [28].

In this paper, we consider non-aggregate data collection problem in WSNs. Among papers in non-aggregate data collection, researchers usually investigate how fast information can be collected from all sensor nodes to base station over tree-based structure. They have proposed numerous scheduling algorithms with the aim of minimizing the number of time slots required to complete data collection process under various scenarios and assumptions. As an example, Zhao and Tang [4] design a schedule algorithm based on colouring approach, in which the number of used colour is equal to the number of used time slot for data collection. Moreover, schedule algorithm is also effected by different assumptions of the problem like buffer limitation, battery lifetime etc. For example, Yeoum *et al.* [30] also proposed a scheduling algorithm for data collection by which the buffer of each sensor node contains at most two data packet in a whole data collection process. In [8], there are several solutions such as time slot assignment, channel assignment scheduling, tree-construction algorithms in data collection are provided and discussed solidly for solving scheduling problem either in non-aggregate or aggregate scenarios. Similarly, Chen *et al.* [9], [10] study the lower bounds of data collection delay in arbitrary networks. They are considered as state-of-the-art schemes for solving non-aggregate data collection problem for arbitrary WSN.

However, those efforts have only been investigated for static traffic where the set of reporting nodes remain unchanged over sampling interval. As mentioned in introduction section, a schedule designed for a light traffic is not suitable for a heavy traffic and vice versa. Therefore, above studies are inefficient in dynamic traffic scenario due to extra energies and latency overheads of identifying the new traffic and constructing new schedules over the sensor network. Zhao and Tang [4] proposed a solution for data collection problem with dynamic traffic in WSNs. The authors first assume all sensor nodes have data to sent to the base station and design a scheduling algorithm to assign collision-free transmission slot for them. Note that the salient feature of TPO schedule is that it enables all available data are always sent in subsequent scheduled transmission slots starting from the first one. The authors then proposed an adaptive traffic mechanism to exploit that feature and makes sensor nodes self-adapt to current data traffic. Nevertheless, this proposed schedule leads to much unnecessary delay due to its tightly constrain —the parent nodes have to send their data later than its descendant even that constrain is only required for aggregate-data collection. To solve the problem, we introduce a novel scheduling scheme which obtains better data collection delay for full traffic and also works effectively in dealing with dynamic traffic scenario.

III. Preliminaries

A. Assumptions

A given sensor network is modeled as a graph $G = (V, E)$, where the set of sensor nodes is represented by the set of vertices V in G, and the set of communication links between nodes is represented by the set of edges E in G. A communication link between two nodes exists if they are located within the transmission range of each other. Assume that there is a single base station (BaseStation) in the network, which collects sensory data from sensor nodes once per sampling interval. The same as [16] and [17], sensor nodes are assumed to be organized in a data collection tree $T(\text{BaseStation}) = (V_T, E_T)$ rooted at the base station BaseStation, where $V_T = V$ and $E_T \subseteq E$.

For each sampling interval, a sensor node is assumed to generate one packet of sensory data with a certain probability and transmit it to the BaseStation in a multi-hop manner. Time is partitioned in timeslots, indexed from 1 to m, where m is the last timeslot BaseStation may still receive some data from its children. Obviously, m is the total number of timeslots required for collecting all the data at BaseStation; and it reflects the delay of the data collection process. Each timelot is long enough for a node to either send or receive a single data packet. Clocks of sensor nodes are assumed to be synchronized by a separated protocol. A node has a buffer with sufficient capacity for storing all received data before being able to forward the data to the next hop.

For non-aggregated data collection, besides reporting its own data, each internal node of a data collection tree has to forward all received data from its descendants to its parent without any aggregation process. For the sake of simplicity, we use the same collision model as in [4], according to which a collision occurs when a node sends its data at the same time as any other node within its 2-hop neighbors in the tree. The set of 2-hop neighbors of a node v is called the collision set of v, denoted by $CS(v)$. We summarize the frequently used notations in the Table I.

B. Problem Formulation

In a given data collection tree T, each node v requires multiple sending timeslots to send its own data and to forward data of its descendants to its parent, if any. To guarantee a successful data collection at the BaseStation without any data loss, the number of required sending timeslots of a node must be sufficient and the schedule must be collision-free. Accordingly, the set of sending timeslots $ST(v)$ allocated for

TABLE I
FREQUENTLY USED NOTATIONS

Notation	Description
BaseStation	The base station in the network
V	Set of vertices (sensor nodes) in the network
E	Set of edges (communication links between nodes) in the network
$T(v)$	The tree T rooted at node v
$Pr(u)$	Parent of node u in a tree
$Ch(v)$	Set of children of node v in a tree
$ST(v)$	Set of sending timeslots of node v
$CS(v)$	Collision set of node v (containing 2-hop neighbors of node v)

node v should follow the constraints below.

$$\left|ST(v)\right| \geq \sum_{x \in Ch(v)} \left|ST(x)\right|, \quad \forall\, u \in V \tag{1}$$

$$ST(v) \cap \bigcup_{x \in CS(v)} ST(x) = \phi, \quad \forall\, u \in V \tag{2}$$

The objective of the problem is to find a schedule for all nodes in a given data collection tree to collect and transmit the sensory data to the base station such that the delay of data collection is minimized. A solution for the problem should be applicable in two different cases, full data traffic and dynamic data traffic. In full data traffic, every node has some data to send, while only some random nodes have data to send in the case of dynamic data traffic. For the case of full data traffic, every leaf node of the data collection tree is scheduled exactly one timeslot for sending its own data. It means the following constraint should also be satisfied when solving the problem.

$$\left|ST(v)\right| = 1, \quad \forall\, v \in V \wedge Ch(v) = \phi \tag{3}$$

Without loss of generality, the objective of the problem under the condition of full data traffic can be restated as the minimization of the last timeslot m until which the base station should wait to receive data. It is apparent that m depends on the schedule $ST(v), \forall v \in V$. Therefore, the problem can be formulated as follows.

$$\begin{aligned} &\min \ \max_{v \in V}(ST(v)) \\ &\text{subject to: } Equation\ (1) \quad \text{and} \quad Equation\ (2) \\ &\qquad\qquad \text{and} \quad Equation\ (3) \end{aligned} \tag{4}$$

Consider an example illustrating the problem given by (4). With the given data collection tree shown in 1(a), there are two different solutions to the problem presented in 1(b) and 1(c). As observed, under the condition of full data traffic, the delay of the data collection process given by the schedule A as in 1(b) is 16, while that of the schedule B as in 1(c) is 15. The improvement is achieved by allowing node A to transmit its data packet at early as possible (at timeslot 1 in this case) instead of waiting to receiving and forwarding the data from its children first. As a result, the schedule B is selected as the best solution for the problem.

Because the change of data traffic is unpredictable, the solution to the problem for full traffic is desirable in any circumstance. To achieve the goal of minimizing the delay under the condition of dynamic traffic, it is necessary to identify the timeslots at which a node finishes its data transmission. Such an identification enables the base station to terminate the collection process as earlier as possible to shorten the delay, and also enables internal nodes to save energy by putting the nodes in the sleeping mode.

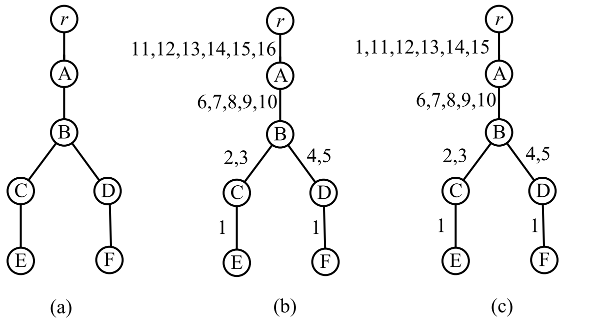


Fig. 1. Scheduling for a given data collection tree. The number attached to each edge represents the sending timeslot(s). (a) A simple topology. (b) A valid schedule A. (c) A valid schedule B.

Consider the two data collection schedules as shown in Fig. 1. Under the condition of dynamic traffic, assume that there are half of nodes in the network have data to send. More specifically, only three nodes A, C, and F are assumed to have data to transmit toward the base station. According to the two schedules shown in Fig. 1, the transmissions are presented in Fig. 2. With the schedule A and under the dynamic traffic condition, the 2(a) shows that BaseStation receives no data at timeslot 14; hence, it will finish the data collection process at the end of this timeslot. It means the delay of the schedule A is 14 timeslots. A better result is obtained by applying the schedule B under dynamic traffic. It achieves the delay of 13 timeslots as shown in 2(b). Furthermore, node B is put in sleeping mode at timeslots 9 $\sim$ 10 in both two cases to save energy after receiving no data at timeslots 3 and 5.

IV. THE PROPOSED SCHEME

A. Overall Approach

Without loss of generality, assume that the set of sending timeslots of a node is in ascending order. Let $tx_i(v)$ denote the i^{th} transmission slot in the set of scheduled sending timeslots $ST(v)$, i.e. $t_i(v) \in ST(v), 1 \leq i \leq |ST(v)|$. It can be drawn a characteristic of the scheduling method A presented through the 1(b) as follows.

$$t_i(v) > t_i(u), \quad \forall\, u \in Ch(v) \tag{5}$$

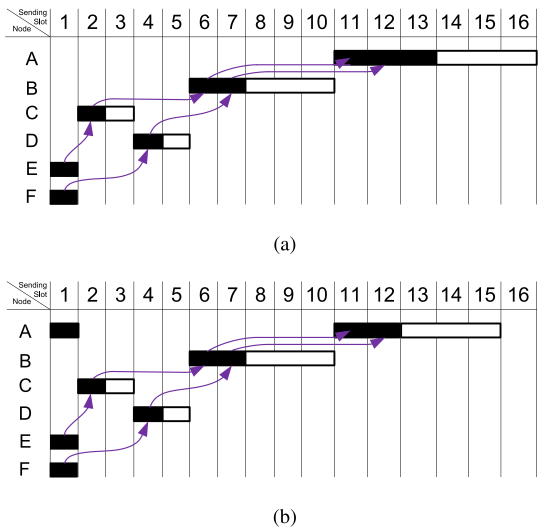


Fig. 2. Schedule of nodes in case of dynamic data traffic. The numbers denote the sending timeslots. A black cell represents a timeslot at which a node is scheduled to transmit its data, and a white one represents a timeslot at which a node does not have any data to send. (a) The result obtained by applying schedule A. (b) The result obtained by applying schedule B.

In fact, the scheduling method A was the key idea of the best-known TPO scheme [4], which has been the best-known solution to the dynamic traffic data collection problem (4). TPO applied the method A with the aim at the utilization of successive timeslots for nodes to transmit their data. With successive transmitting timeslots, it is easy to realize that if a node has no data to send at a certain timeslot, it will never transmit any data at later ones. This is the stop condition which empowers a node to terminate its tasks of data collection and transmission earlier (within a sampling interval) for energy saving in the case of dynamic data traffic. However, it is observable that the proposed method A introduces a higher delay in data collection than the method B because of the utilization of successive timeslots.

Our DETA scheme strikes this deficient out by enabling a node to collision-freely send its buffered data as early as possible to reduce the delay. In addition, DETA deals with the dynamic data traffic by a more adaptive mechanism. Thanks to the smaller delay in data collection, the adaptive mechanism is also able to enable nodes to terminates its data collection process earlier according to the data traffic to reduce the delay and save the energy. As a result, DETA is both delay and energy-efficient.

The proposed scheme, DETA, is designed with two phases as follows:

- **Phase 1. Data Collection Scheduling:** constructs a schedule for each node to transmit data toward the base station under the condition of full data traffic.
- **Phase 2. Adaptive Data Transmission:** defines a criterion based on which a node can adaptively stop its data collection task in a sampling interval in accordance with the dynamic data traffic.

B. Data Collection Scheduling

Under the full data traffic, every node has a data packet to send. Therefore, DETA scheduling algorithm has to produce a *full schedule* for every node in the network to transmit its own data and recursively forward the data of its descendants in the tree rooted at the node. The algorithm works in iterations until every node is allocated enough transmitting timeslots to accomplish its task. More specifically, the number of allocated timeslots for a node υ must satisfy the constraint (1).

For each iteration, the algorithm starts with the set of nodes whose descendants are fully allocated timeslots for data transmissions. Such a node is called a *ready node*. It means initially all the leaf nodes of the data collection tree are scheduled. Each node is assigned only one timeslot per iteration. A timeslot allocated for a node υ is following two principles: (i) the timeslot for transmitting the data packet of υ can be selected from 1; and (ii) the timeslot for forwarding the data packet generated by a descendant of υ has to be after the timeslot at which υ receives the packet. Note that timeslot assignment for nodes must satisfy the constraint (2). For a formal presentation, allocable timeslots for a node are categorized into two types as described below.

1) The first type of timeslot (*type*-1) is assigned to a node for transmitting the data it generates in a sampling interval. Each node requires only one transmission timeslot of this type per interval. The algorithm assigns such a timeslot to the node if it is a ready node in a particular iteration. Because data generated by the node is in place, the transmission (timeslot) for the data should be scheduled as early as possible with the condition that no collision occurs. In the iteration k, the sending timeslot $st_k(\upsilon)$ of this type is given by:

$$st_k(\upsilon) \leftarrow \min\{\ t \mid t > 0 \\ \wedge\ t \notin ST(\upsilon) \cup \Big(\bigcup_{y \in CS(\upsilon)} ST(y) \Big)\ \} \quad (6)$$

2) The second type of timeslot (*type*-2) is used for a node to forward data from its descendants to its parent in the data collection tree. Since a node is responsible for forwarding the data of all its descendants, the number of timeslots of this type demanded by the node is equal to the number of its descendants. For each iteration, only one timeslot is allocated for a node, and it has to be greater (later) than the one assigned to any child of the node. Note that a node may have multiple children. In case all children of a node υ have acquired all demanded timeslots at iteration k, the timeslots allocated for node υ at iterations $i > k$ should be greater than $\max\{st_k(x), \forall x \in Ch(\upsilon)\}$. Accordingly, the sending timeslot of node υ in the iteration $i \geq k$, $st_i(\upsilon)$, of this type is given by:

$$st_i(\upsilon) \leftarrow \min\{\ t \mid t > \max\{st_k(x),\ \forall x \in Ch(\upsilon)\} \\ \wedge\ t \notin ST(\upsilon) \cup \Big(\bigcup_{y \in CS(\upsilon)} ST(y) \Big)\ \} \quad (7)$$

It is obvious that to obtain a full schedule, the number of iterations performed for a node should be equal to the number of its descendants plus one. The first iteration is to allocate a type-1 timeslot for each leaf node to send its generated data packet, and to allocate type-2 timeslots for other nodes

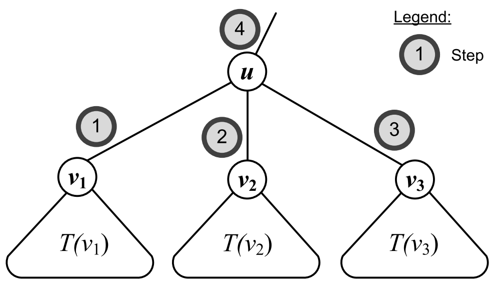


Fig. 3. An example execution of DETA.

to forward these data packets to the parent nodes. Timeslots allocated in the first iteration are call *anchor timeslots*. The anchor timeslot is the point at which an internal node is supposed to have (in its buffer) the data of its farthest descendant in the subtree rooted at the node if any. Data sent by other descendants should arrive at the node earlier. Therefore, every internal node should wait to receive data until its anchor timeslot regardless of the traffic pattern before performing the adaptive mechanism presented in subsection IV-C. In later iterations, a node is able to allocate an earlier timeslot than the anchor timeslot for transmitting any data which is stored in its buffer. The timeslot assigned to a node in the last iteration is assumed to be used for transmitting data generated by the node. In general, with the assumption that nodes adopt the store-and-forward technique in data communication, after being establishing a full schedule, a node can send a data packet stored in its buffer at any timeslot in its schedule.

We first assume that each sensor node has one data packet to send to the base station and use an example shown in Fig. 3 to explain the algorithm. Besides recycling the terminologies introduced in the previous sections, we present some new notations:

— $p(u)$: parent node of u
— $buff(u)$: the number of available data packets in the memory of u
— $cnt(u)$: the number of assigned transmission slot to node u in a certain schedule round

More specifically, in the first iteration, considering node u has three children v_1, v_2 and v_3, as in Fig. 3, and node u is only scheduled after all these children nodes $-v_1, v_2$ and v_3- have been scheduled.

So, the scheduling process starts from the smallest sub-trees, where only the leaf nodes and its parent are included. The parent node is scheduled right after all its children have been finished their schedule.

The procedure is repeated upstream towards to the sink. In the next iteration, the nodes that have been assigned enough the required time slots will not be evolved. Algorithm starts a new iteration from the node that all its children have been finished the schedule, the same operation as in the first iteration. Algorithm stops when all the children of the sink are assigned enough the required number of transmission slots. The pseudo-code of DETA schedule is provided in Algorithm 1 and 2.

There are two types of time slot that a sensor node can be obtained in each schedule iteration.

Two types of scheduled time slots are assigned to a node u by Sch-node(u) algorithm. As mentioned, the first type is a single value which greater than the maximum obtained time slots of all its children as in step 1 of algorithm 3. The second type of time slot contains all reuse values of time slot outside collision zone which enable sensor to send all data in its buffer as in steps $4 - 14$ of Algorithm 3. The amount of assigned time slots is equal to the number of data packets that the node has to send. This type of time slot is always smaller than the last obtained value of type 1. Note that the time slots belonged to type one is always possible to obtain whenever a node is involved in a certain scheduling iteration, but the second type sometimes cannot be obtained due to collision constraints and the limited number of required time slots reserved by each sensor node.

Algorithm 1 Aggregation-Tree-Traversal

```
Input: $Tr = \{p(v), Ch(v), cc(v), \forall v \in Tr\}, root, L$
Output: $L$
1: for each $v \in Ch(root)$ do
2:     Aggregation-Tree-Traversal($Tr, v, L$)
3:     $cc(root) \leftarrow cc(root) + |Ch(v)|$
4: end for
5: $L \leftarrow L \cup \{root\}$
```

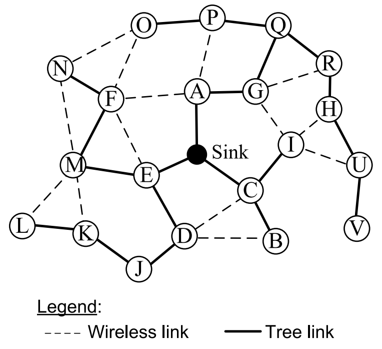


Fig. 4. An example network topology.

Considering a example network topology as in Fig. 4, and Fig. 5 illustrate the results obtained by DETA and TPO for this network topology. Note that we do not consider dynamic traffic scenario at this moment, thus the data collection delay is reflected by the maximum value of used time slots, which are 30 with TPO schedule and 25 with DETA schedule.

C. Adaptive Network Traffic Mechanism

Note that Algorithm 2 produces a full schedule for every node in the network. Under the dynamic traffic condition, a node may receive data at just some timeslots in its full schedule. Hence, it may not need to be active for data reception till the last timeslot of the schedule. This subsection discusses the adaptive mechanism that enables every sensor node to detect the completion of its mission of data collection. The objective of such a detection is to minimize both the delay and the number of transmissions of the whole data collection process.

As per the discussion in subsection IV-B, the anchor timeslot of each node is supposed to use for forwarding the data

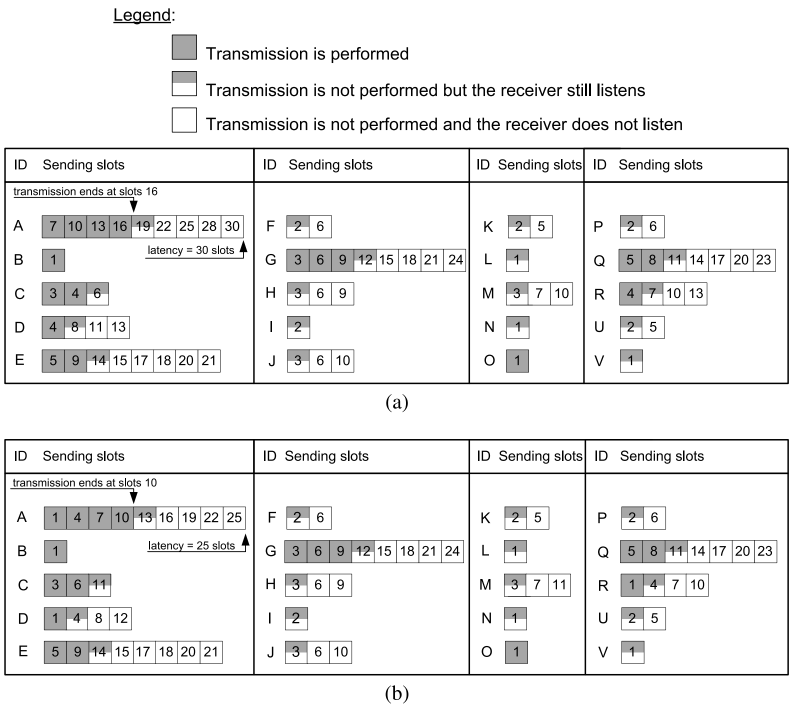


Fig. 5. An example execution of scheduling results from DETA and TPO algorithms for the network topology in Fig. 4. (a) The results by applying TPO scheme for the network in Fig. 4, the latency of data collection for full traffic (all nodes have data to report) and dynamic traffic (only nodes A, B, C, D, E, G, Q and R have data to report) are 30 slots and 19 slots, respectively. (b) The results by applying DETA scheme for the network in Fig. 4, the latency of data collection for of full traffic (all nodes have data to report) and dynamic traffic (only nodes A, B, C, D, E, G, Q and R have data to report) are 25 slots and 14 slots, respectively.

Algorithm 2 DETA-Scheduling

```
Input: $Tr = \{p(v), Ch(v), cc(v), \forall v \in Tr\}, root$
Output: $ST(v), \forall v \in Tr$
1: $L \leftarrow \emptyset$
2: for each $v \in L$ do
3:     $cc(v) \leftarrow 0$
4:     $ST(v) \leftarrow \emptyset$
5: end for

6: Aggregation-Tree-Traversal($Tr, root, L$)

7: while $L \neq \emptyset$ do
8:     for each $v \in L$ do
9:         if $|ST(v)| < cc(v)$ then
10:            $slot \leftarrow \min\{ t \mid t > \max\{\text{last-scheduled-slot}(ST(x)), \forall x \in Ch(v)\} \wedge t \notin ST(v) \cup \left(\bigcup_{y \in CS(v)} ST(y)\right) \}$
11:        else
12:            $slot \leftarrow \min\{ t \mid t > 0 \wedge t \notin ST(v) \cup \left(\bigcup_{y \in CS(v)} ST(y)\right) \}$
13:            $L \leftarrow L \setminus v$
14:        end if
15:        $ST(v) \leftarrow ST(v) \cup \{slot\}$
16:    end for
17: end while
```

of the farthest descendant on the subtree rooted at the node. In case the farthest node has no data to send, at anchor timeslot the node will forward any other data which have been stored in its buffer if any. It is worth noting that the data stored in the buffer could be of other descendants or the node itself. In other words, at anchor timeslot if there is no data buffered at the node, all the node's descendants and itself have no data to send.

As a result, with the full schedule provided by Algorithm 2, under a dynamic traffic pattern, a node can conclude its data collection at any timeslot starting from the anchor one onward. At the anchor timeslot, if a node has no data in the buffer to send, the node will no longer receive and send any data. Based on the discussion, the mechanism of adaptive data transmission is designed, in which every node performs a check on its buffer for any data as of the anchor timeslot. If the buffer is empty, the node may immediately switch to the sleep mode in the current session of data collection to save energy. Moreover, with this mechanism, the base station can conclude the data collection earlier, thereby improving the total delay.

We describe an adaptive mechanism that enables sensor nodes to detect the completion of data transmissions in their sub-tree. Note that in the same round of the DETA schedule, the first type time slot which is assigned to the parent is always greater than of all its children. Moreover, the first type of transmission slot that is assigned to a node is greater than the remaining type in each schedule round. Therefore, if the parent node v does not receive any data from node x at any $st_1(x) \in ST_1(x)$, and $st_1(x)$ is greater than the maximum value in $ST_2(x)$, node v can go into sleep mode in all subsequent receiving transmission slots in $ST_1(x)$. The base station, instead of listening until the end of schedule, concludes data collection once it infers that all its children have finished transmission, thereby reducing collecting delay

For example, suppose that only nodes A, B, C, D, E, G, Q and R generate packets to send. The actual transmissions occur as in Fig. 5(a) and Fig. 5(b) for the DETA and the TPO scheme, respectively. It is clear that our approach can significantly reduce the number of transmission slots for data collection according to the change of traffic. The DETA scheme spends only 10 transmission slots to collect all available data instead of the 16 transmission slots spent by TPO. As a result, base station S concludes that the data collection process has completed at the end of time slot 14 when no data comes from E. However, if we apply the TPO scheme, base station S can only conclude data collection until the end of time slot 19 when there is no data coming from node A.

Proof: Before discussing about the correctness this mechanism, we review again the way of time slots selection. There are two types of transmission slots, $ST_1(v)$ and $ST_2(v)$. In the same round,

- $ST_1(v) \leftarrow ST_1(v) \cup \{st_1(v)\}$, where $st_1(v) \leftarrow min\{r | r > 0, r > st_1(x), \forall x \in Ch(v), r \notin ST(CS(v))\}$.
- $ST_2(v) \leftarrow ST_2(v) \cup \{z\}$, where $\forall z < st_1(v)\}$.

The idea of this mechanism is that since $st_1(x)$ is the upper bound when we try to find $ST_2(x)$. Therefore, if there is no data come to v at $st_1(x)$, it implies that x has no remaining data to send to v. Note that the temporal order of transmission slots of node x is: $z^1(x) < z^2(x) < z^3(x) < \ldots < st_1^1(x) < z^{i-1}(x) < z^i(x) < st_1^k(x) < st_1^{k+1}(x) < \ldots < st_1^j(x)$, where $z \in ST_2(x), st_1(x) \in ST_1(x)$. Therefore, it is guaranteed that if there is no data to send at time slot $st_1(x)$, node x would not have anything to send to its parent v after $st_1(x)$. Then, v can go to sleep mode at all subsequent receiving transmission slots in $ST_1(x)$ after $st_1(x)$. Hence, the correctness of adaptive traffic mechanism is proven.

D. Distributed Implementation of DETA

In this section, we describe how to implement DETA scheduling algorithm in a distributed manner. This algorithm is considered as a distributed implementation of Algorithm 2, thus there is no different from the results by applying two approaches. We will not distinguish between these two version of DETA in the rest of this paper. We first assume that each sensor node u knows its parent $p(u)$, its children $Ch(u)$, its grandparent $gp(u)$ and the size of its sub-tree rooted based on the information from the tree construction step (not discuss in this paper). In distributed DETA algorithm, each sensor node u keeps the following local variables:

— $ST(u)$ contains all the set of sending time slots assigned to node u. Initially, $ST(u) = \phi$.
— $STC(v)$ contains the sending time slots of each child $v \in Ch(u)$. Initially, $STC(v) = \phi, \forall v \in Ch(u)$.
— $CS(u)$ contains all the nodes that conflict with v and have not been assigned the required numbers of transmission slots. Initially, $CS(v) = \phi$.
— $buff(u)$ is the number of available data packets in the memory of node u.
— $cnt(u)$ is the number of assigned transmission slot to node u in a certain schedule round.

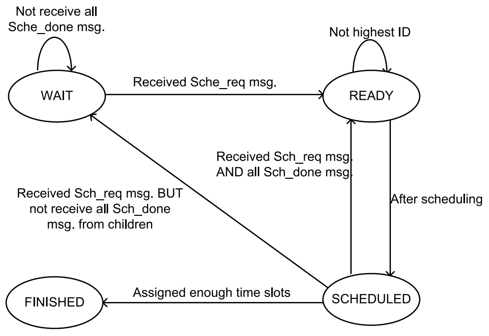

Not receive all Sche_done msg.
Not highest ID
WAIT
Received Sche_req msg.
READY
Received Sch_req msg. AND all Sch_done msg.
Received Sch_req msg. BUT not receive all Sch_done msg. from children
After scheduling
FINISHED
Assigned enough time slots
SCHEDULED

Fig. 6. State diagram of DETA Algorithm.

Algorithm 3 Distributed-DETA-Sch(s): Algorithm Executed by the Base Station s

```
1: s sends a Sch_req message
2: while $\exists u \in Ch(s)$, State(u) $\neq Finished$ do
3:     if s receives all Sch_done msg. its Ch(s) then
4:         s sends new Sch_req msg.;
5:     end if
6: end while
```

We define four states that the sensor node u can be kept during scheduling process, as in Fig. 6:

- *Ready*: all children of node u have finished their schedule.
- *Wait*: there is a children or descendant of node u that does not finish its schedule.
- *Scheduled*: node u has just been assigned the time slot in $Ready$ state.
- *Finished*: node u has assigned enough the amount of its required time slot.

We design two algorithms for two targeted sensors nodes in the network: sink node and reporting node, and they elaborate each others to make the scheduling algorithm works into iterations. For the sink node, its duty is to initialize the scheduling process as well as identifies the end of that process. The sink first generates a Sch_req message and broadcasts to all its neighbour to inform that the algorithm has been started. The Sch_req message has the format $\langle serial, tag \rangle$, where $serial$ is the number represented for the index of current iteration, tag is a value to distinguish this control message with others. The pseudo-code of DETA distributed algorithm for the sink node can be shown in Algorithm 3

For other node u, all the leaf nodes are in $Ready$ state, other nodes are in $Wait$ state at the first iteration. When node u receives the Sch_req message from the sink, there are a set or subsequent procedures as:

- When a node u in $Ready$ state, if it receives the Sch_req from its parent, it will check that whether it has highest ID compared with other nodes in $CS(u)$ which are in $Ready$ state or not. If so, it will select the sending time slots. After selecting its sending time slots, the node will broadcast the Sch_done message to two-hop neighbors. The format of Sch_done message will be presented later.
- When node u is in $Wait$ state, if it receives Sch_req message from its parent it checks whether receives all the

Sch_done messages from all of its children, considering the children who did not finish the schedule. If so, its state will change to $Ready$. Otherwise, node u keeps it at $Wait$ state and then forwards Sch_req message to all its children.

- When node u is in $Scheduled$ state, if it receives Sch_req message from its parents, it will change its state to $Ready$ if all of its children are in $Finished$ state. Otherwise, its state will be changed to $Wait$ after receiving Sch_req message.

The algorithm stops when the base station detects that all of its children have finished their scheduling by observing the received Sch_done message. After finishing scheduling process, each sensor node u would know its sending slots $ST(v)$ and its receiving time slots $STC(v), \forall v \in Ch(u)$. The pseudo-code of DETA distributed algorithm for each node u can be shown in Algorithm 4.

The Sch_done message sent by u has the form of Sch_done$\langle ST_1(u), ST_2(u), buff(u), cnt(u), src, tag\rangle$, where $ST_1(u), ST_2(u), buff(u)$, and $cnt(u)$ have been described above; src the ID of node that sent this message; tag is a boolean value which indicates whether the receiving node has to continue broadcasting this Sch_done message or not. Initially, the value of tag is true. When a node receives Sch_done message from its neighbours with the tag value is true, it will change tag value to false and broadcast this message to all its neighbours. If a node receives Sch_done message with tag is false, it will not broadcast this message any more.

Algorithm 4 Distributed-DETA-Sch (u): //Algorithm Executed by a Sensor Node u

```
1: $buff(u) = 1$
2: if $u$ is a leaf node then
3:     State($u$) ← $Ready$;
4: else
5:     State($u$) ← $Wait$;
6: end if
7: while $|ST(u)| < d(u)$ do
8:     switch $State(v)$ do
9:         case $Ready$
10:            if $u$ receives Sch_req msg. then
11:                if $ID(u) > ID(v), State(v) = Ready, \forall v \in CS(v)$ then
12:                    $cnt(u) = 0$
13:                    Sch-node($u, buff(u), cnt(u)$);
14:                    State($u$) ← $Scheduled$;
15:                    $u$ broadcasts Sch_done msg.;
16:                end if
17:            end if
18:        case $Wait$
19:            if $u$ receives Sch_req msg. then
20:                if $\exists v \in Ch(u)$, State($v$) $\neq Finished$ then
21:                    $u$ broadcasts Sch_req msg.;
22:                end if
23:            end if
24:            while $u$ receives Sch_done msg. from $v$, $v \in Ch(u)$ do
25:                $STC(v) = ST(v)$;
26:                $buff(u) = buff(u) + cnt(v)$;
27:                if $State(v) = Scheduled|Finished, \forall v \in Ch(u)$ then
28:                    State($u$) ← $Ready$;
29:                end if
30:            end while
31:        case $Scheduled$
32:            if $u$ receives Sch_req msg. then
33:                if State($v$) $= Finished, \forall v \in Ch(u)$ then
34:                    State($u$) ← $Ready$;
35:                else
36:                    State($u$) ← $Wait$;
37:                end if
38:            end if
39: end while
```

V. Performance Evaluation

A. Simulation Environment

We develop a simulation to illustrate the effectiveness and superiority of the DETA scheme compared with the existing schemes. We simulate WSNs with two different shapes of sensing area: square and rectangle. In reality, the square shape can be used for the monitoring applications of WSNs over a large area such as a city and a military zone, the rectangle shape is for simulating the border monitoring applications like barriers or boundaries monitoring. The characteristics of sensor node are remained unchanged during the simulation, each sensor node have identical transmission range of 15m. In the data collection process, when a receiving node is scheduled to receive data from a sending node, there is no other node which is allowed to be scheduled for sending data to that receiving node in the same time slot. Sensor node wakes up at its scheduled transmission slots to send and receive data and goes to sleep after finishing its transmission. We conduct a Breadth-First Search (BFS)) tree rooted at the base station for data collection. Even though there are many other ways to construct better structures for data collection like [16] and [17], since our primary objective is to design an efficient scheduling strategy which is independent to network structure, BFS tree makes evaluation process much simpler and more effective.

1) Sensing Area: We first randomly distribute 100 nodes in a square region of 100m $\times$ 100m. An example network topology and corresponding data collection tree in this simulation has been shown in Fig. 7. We then evaluate the performance of four state-of-the-art schemes DETA, BFA [10], FDC [7], TPO [4]. As discussed, network traffic is dynamically changed over sampling intervals. To implement this, we randomly generate 100 sets of nodes whose size varies from 1 to 100 at the beginning of each sampling interval. Each set contains a random number representing the IDs of nodes that do not have data to send to the base station at that sampling interval, and data collection process only conduct for the rest nodes.

Moreover, one of the salient features of DETA is that it enables parallel transmissions between the internal nodes and

TABLE II
FREQUENCY USE SYMBOLS

Symbols	Descriptions
$\xi_{\text{total}}/P_{\text{total}}$	energy/power consumed in a whole data collection process
$\xi_{\text{setup}}/P_{\text{setup}}$	energy/power consumed in set-up steps, $\xi_{\text{setup}} = 1895J$
$\xi_{\text{tr\_rx}}/P_{\text{tr\_rx}}$	energy/power consumed for transmit and receive data
$\xi_{\text{idle}}/P_{\text{idle}}$	energy/power consumed for waiting to receive data from children but no data come
$\xi_{\text{sleep}}/P_{\text{sleep}}$	energy consumed of a node when it is not active
$t_{\text{tr\_rx}}$	time interval of sending and receiving data
t_{idle}	time interval of waiting to receive data from children but no data come
t_{sleep}	time interval of a node when it is not active
t_{interval}	time interval of each sampling

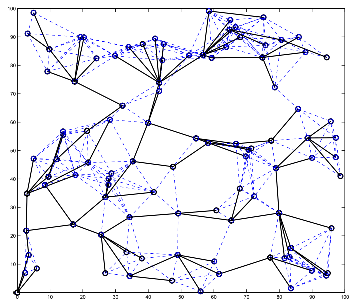


Fig. 7. An example square sensing area and its corresponding data collection tree.

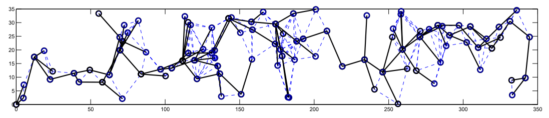


Fig. 8. An example rectangle sensing area and its corresponding data collection tree.

its descendants to reduce data collection delay as presented in the previous presentation, therefore the DETA is more suitable for WSNs in border monitoring applications. We evaluate the performance of our scheme over a sensing area with rectangle shape where the collection paths are usually longer than that in the square shape. To simulate this environment, we vary the length:width ratio of sensing area from 5, 10, and 15.

2) Energy Model: We conduct energy model based on Bluetooth 4.1 standard [27], and similar with energy model as in [4] and [26]. We first introduce some variables that would be used in this presentation as in Table II.

The value of ξ_{total} can be calculated as:

$$\xi_{\text{total}} = \xi_{\text{set up}} + \xi_{\text{tx\_rx}} + \xi_{\text{idle}} + \xi_{\text{sleep}}, \quad (8)$$

Since $\xi = t.P$, t is duration time and P is power consumption, the E.q. (13) can be rephrased as:

$$\xi_{\text{total}} = \xi_{\text{set up}} + \{t_{\text{tx\_rx}}.P_{\text{tx\_rx}}\} + \{t_{\text{idle}}.P_{\text{idle}}\} + \{(t_{\text{interval}} - t_{\text{tx\_rx}} - t_{\text{idle}}).P_{\text{sleep}}\}, \quad (9)$$

In addition, to support for above calculation, the Table III summarizes the values of variables calculated from [17], [26]. We assume that each time slot has a length of 625μs.

TABLE III
VALUES OF VARIABLES

Symbols	Values
Voltage	3.3V
$P_{\text{tx\_rx}}$	35mA × 3.3V = 115.5mW
$t_{\text{tr\_rx}}$	$ST_{\text{transceive}} \times 6\mu$s
P_{idle}	0.75 × 115.5 = 86.6mW
t_{idle}	$ST_{\text{idle}} \times 6\mu$s
P_{sleep}	4mA × 3.3V = 14mW
t_{sleep}	$ST_{\text{sleep}} \times 6\mu$s
ξ_{setup}	462 J

B. Simulation Results

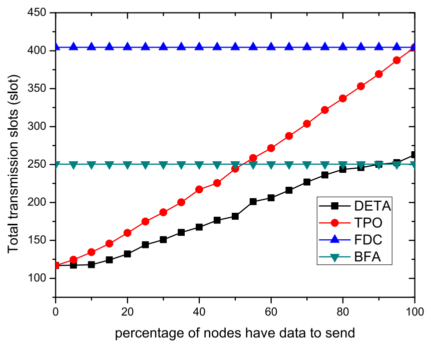
1) Results of Square Sensing Area: We have experienced with many random network topologies and have obtained similar trends of performance. In this experiment, for each set of parameter settings, the reported results are the average of 200 runs on random network topologies.

Fig. 9(a) and Fig. 9(b) show the data collection delay and energy consumption of DETA, TPO [4], FDC [7], and BFA [10] when network traffic varies from 100% nodes to no node that has data to send in each sampling interval. The collection delay of the DETA is always smaller than the other schemes. As FDC and BFA are designed for the static network traffic, they are not able to adapt to the drastic change of traffic over different intervals. They become inefficient when the traffic is lighter, especially since the number of nodes have data to report is less than 80%.

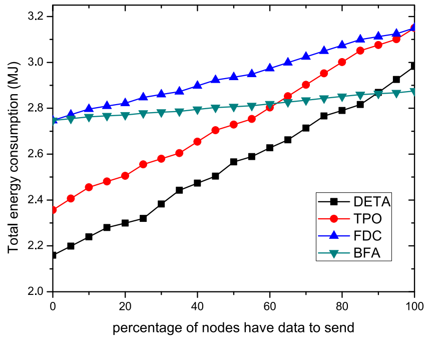
The DETA achieves up to 32.4% improvement in terms of data collection delay compared with the TPO. A disadvantage of the DETA is that we spend a little more energy for idle listening when some nodes do not have their own data to send at re-use transmission slots. However, because we used smaller amount of time slot for data collection process, the sensor node could go to sleep earlier and save energy of sleep state. Even the energy consumed in sleep state smaller than idle listening but if it would be considerable if all sensor nodes in the network can save certain amount of sleep energy right after it finished data transmission at the last scheduled time slot. Fig. 9(b) also shown that we obtain up to 5.4% improvement of energy consumption compared with the TPO.

2) Results of Rectangle Sensing Area:

a) Impact of the Sensing Area: As can we seen from the previous simulation, the FDC [7], and BFA [10] are not candidates for solving dynamic traffic data collection problem, thus in this section we only simulate the TPO [4] together

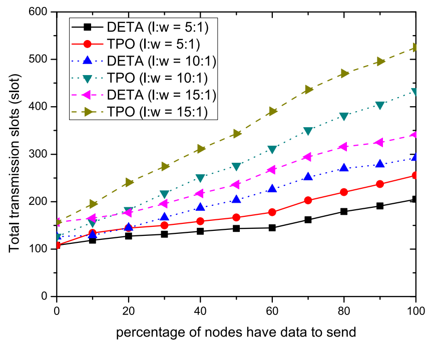


(a)

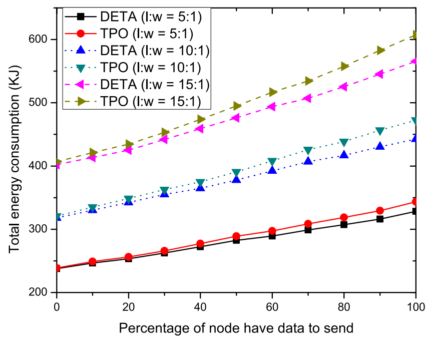


(b)

Fig. 9. Results of varying traffic patterns. (a) Data collection delay. (b) Energy consumption.



(a)



(b)

Fig. 11. Varying traffic from 0% to 100% nodes have data to send. (a) Data Collection Delay. (b) Energy consumption.

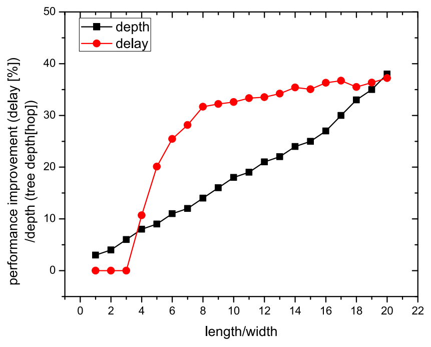


Fig. 10. Performance improvement compared with TPO when varying the area of sensing field.

with DETA to evaluate their performance. The simulation verifies the performance of DETA and TPO when varying the lengh:width ratio of the sensing area and the varying of data traffic. In this parameter settings, we assume that all sensor nodes have data to report to the base station and run TPO and DETA algorithm for different sensing area with length:width ratio vary from 5, 10, 15, and 20. Each simulation runs through 200 network topologies. The Fig. 8 shown an example of network topology and corresponding data collection tree with the length:width ratio is 10. The results of simulation is presented in the Fig. 10. As can be seen from the Figure, our proposed has significant improvement in terms of data collection delay when the depth of data collection tree is greater than 5. The DETA continues achieving better results along to the increment of the depth of collection tree up to 37%.

3) Impact of Dynamic Traffic: To verify the impact of dynamic traffic over rectangle data collection area, we fix the length:width is 5, 10, and 15, and then vary the percentage of nodes have data to send from 0% to 100% in each of those settings. The similar results could be obtained as in the square shape. Nevertheless, since data collection in border line applications has longer depth, thus the DETA could achieve much more improvement in terms of delay. The results of simulation can be seen in Fig. 11(a) and Fig. 11(b). It is clear from the figures that the bigger value of length:width, the better performance DETA scheme could obtain compared with the TPO in either data collection delay and energy consumption. Therefore, one small conclusion could be drawn here, the DETA scheme is more suitable for barriers or boundaries monitoring applications, in that the sensors are deployed over a long distance from the source.

VI. Conclusion

In this paper, we have presented scheduling algorithm to schedule sensor nodes in the network to send their data to the base station with minimum delay. In addition, we designed an adaptive mechanism to allow each sensor node to go to sleep early according to the current data distribution of a sampling interval, thereby reducing data collection delay and energy consumption. The simulation results show that our proposed scheme achieves better performance than the existing schemes. As mentioned, our scheme would be more benefits with boundaries monitoring applications because the amount of sensor nodes that could send data simultaneously is bigger.

References

[1] D. Wang, J. Xu, J. Liu, and F. Wang, “Mobile filtering for error-bounded data collection in sensor networks,” in *Proc. IEEE 28th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2008, pp. 530–537.

[2] N. Xu *et al.*, "A wireless sensor network for structural monitoring," in *Proc. ACM 2nd Int. Conf. Embedded Netw. Sensor Syst. (SenSys)*, Nov. 2004, pp. 13–24.

[3] R. Szewczyk, A. Mainwaring, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *Proc. SenSys*, 2004, pp. 214–226.

[4] W. Zhao and X. Tang, "Scheduling sensor data collection with dynamic traffic patterns," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 4, pp. 789–802, Apr. 2013.

[5] 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.

[6] S. Gandham, M. Dawande, and R. Prakash, "Link scheduling in sensor networks: Distributed edge coloring revisited," in *Proc. IEEE INFOCOM*, Mar. 2005, pp. 2492–2501.

[7] O. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 1, pp. 86–89, Jan. 2012.

[8] Y. Zhang, S. Gandham, and Q. Huang, "Distributed minimal time convergecast scheduling for small or sparse data sources," in *Proc. IEEE RTSS*, Dec. 2007, pp. 301–310.

[9] S. Chen, M. Huang, S. Tang, and Y. Wang, "Capacity of data collection in arbitrary wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 1, pp. 52–60, Jan. 2012.

[10] S. Chen, Y. Wang, X.-Y. Li, and X. Shi, "Data collection capacity of random-deployed wireless sensor networks," in *Proc. IEEE GLOBECOM*, Nov. 2009, pp. 1–6.

[11] T. Liu, T. Li, and Y. Chen, "A distributed TDMA-based data gathering scheme for wireless sensor networks," *IEICE Trans. Inf. Syst.*, vol. E96-D, no. 9, pp. 2135–2138, 2013.

[12] A. Q. Zhao, Y. N. Weng, Y. Lu, and C. Y. Liu, "Research on dynamic routing mechanisms in wireless sensor networks," *Sci. World J.*, vol. 2014, Apr. 2014, Art. no. 165694.

[13] W. Zhao and X. Tang, "Scheduling data collection with dynamic traffic patterns in wireless sensor networks," *Proc. IEEE INFOCOM*, Apr. 2011, pp. 286–290.

[14] C.-T. Cheng, C. K. Tse, and F. C. M. Lau, "A delay-aware data collection network structure for wireless sensor networks," *IEEE Sensors J.*, vol. 11, no. 3, pp. 699–710, Mar. 2011.

[15] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo, "The collection tree protocol," *TinyOS Enhancement Proposals*, vol. 123, pp. 1–7, Dec. 2007.

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

[17] 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 ICDCS*, Jun. 2010, pp. 827–836.

[18] A. Ghosh, O. D. Incel, V. S. A. Kumar, and B. Krishnamachri, "Multi-channel scheduling algorithms for fast aggregated convergecast in sensor networks," in *Proc. IEEE Int. Conf. Mobile Adhoc Sensor Syst.*, pp. 363–372, Oct. 2009.

[19] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Maximum lifetime data gathering and aggregation in wireless sensor networks," in *Proc. IEEE Int. Conf. Netw.*, Aug. 2002, pp. 685–696.

[20] X. Xu, X. Li, X. Mao, S. Tang, S. Wang, and X. Lin, "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.

[21] Y.-C. Kuo and J.-W. Chen, "A power-saving data aggregation algorithm for byzantine faults in wireless sensor networks," *IEICE Trans. Commun.*, vol. E92-B, no. 6, pp. 2201–2208, Jun. 2009.

[22] S. Cho, "Proactive data filtering algorithm for aggregation in wireless sensor networks," *IEICE Trans. Commun.*, vol. E91-B, no. 3, pp. 742–749, Mar. 2010.

[23] E. Lee, S. Park, F. Yu, and S.-H. Kim, "On selection of energy-efficient data aggregation node in wireless sensor networks," *IEICE Trans. Commun.*, vol. E93-B, no. 9, pp. 2436–2439, Sep. 2010.

[24] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 1992.

[25] J. Ma, W. Lou, Y. Wu, X.-Y. Li, and G. Chen, "Energy efficient TDMA sleep scheduling in wireless sensor networks," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 630–638.

[26] J.-C. Cano, J. M. Cano, E. González, C. Calafate, and P. Manzoni, "How does energy consumption impact performance in Bluetooth?" *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 3, pp. 7–9, 2007.

[27] *Bluetooth Specifications*. Accessed: Nov. 2017. [Online]. Available: https://www.bluetooth.org/en-us/specification/adopted-specifications

[28] B. Kang, P. K. Ha, V. Zalyubovskiy, and H. Choo, "A distributed delay-efficient data aggregation scheduling for duty-cycled WSNs," *IEEE Sensors J.*, vol. 11, no. 17, pp. 3422–3437, Apr. 2017.

[29] B. Kang, S. Myoung, and H. Choo, "Distributed degree-based link scheduling for collision avoidance in wireless sensor networks," *IEEE Access*, vol. 4, no. 3, pp. 7452–7468, Sep. 2017.

[30] S. Yeoum, B. Kang, J. Lee, and H. Choo, "Channel and timeslot co-scheduling with minimal channel switching for data aggregation in MWSNs," *Sensors*, vol. 17, no. 5, p. 1030, 2017.

[31] 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.

[32] T. Wang, Y. Li, G. Wang, J. Cao, M. Z. A. Bhuiyan, and W. Jia, "Sustainable and efficient data collection from WSNs to cloud," *IEEE Trans. Sustain. Comput.*, in press.

[33] Y. Yao, Q. Cao, and A. V. Vasilakos, "EDAL: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for heterogeneous wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 810–823, Jun. 2015.

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 Hertfordshire, U.K., in 2015. From 2015 to 2017, he was a Post-Doctoral Researcher with Sungkyunkwan University. He is currently an Assistant Professor with the Department of Data Science, Sejong University. His research interests include cloud computing, IoT, wired/wireless networking, sensor networking, mobile computing, network security protocols, and simulations/numerical analysis.

Phuc Nguyen received the B.S. degree in electronics and telecommunications engineering from the Ho Chi Minh City University of Science, 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 2014. His research interests include delay and energy-efficient scheduling algorithms in wireless sensor networks and wireless communication protocols.

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. Since 1998, he has been with the College of Information and Communication Engineering, Sungkyunkwan University, where he is currently an Associate Professor and a Director with the Convergence Research Institute. Since 2005, he has been a Director of the Intelligent HCI Convergence Research Center (eight-year research program) supported by the Ministry of Knowledge Economy, South Korea under the Information Technology Research Center support program supervised by the Institute of Information Technology Assessment. He has authored or co-authored 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 currently a Vice President with Korean Society for Internet Information (KSII). He has been an Editor-in-Chief of the Journal of KSII for three years and a Journal Editor of the JOURNAL OF COMMUNICATIONS AND NETWORKS, ACM *Transactions on Internet Technology*, *International Journal of Mobile Communication*, and Springer-Verlag *Transactions on Computational Science Journal*, and an Editor of the KSII *Transactions on Internet and Information Systems* since 2006. He is a member of the ACM.