Understanding Multi-Task Schedulability in Duty-Cycling Sensor Networks

Mo Li, *Member, IEEE*, Zhenjiang Li, *Member, IEEE*, Longfei Shangguan, *Student Member, IEEE* , Shaojie Tang, *Member, IEEE*, and Xiang-Yang Li, *Senior Member, IEEE*

Abstract—In many sensor network applications, multiple data forwarding tasks usually exist with different source-destination node pairs. Due to limitations of the duty-cycling operation and interference, however, not all tasks can be guaranteed to be scheduled within their required delay constraints. We investigate a fundamental scheduling problem of both theoretical and practical importance, called multi-task schedulability problem, i.e., given multiple data forwarding tasks, to determine the maximum number of tasks that can be scheduled within their deadlines and work out such a schedule. We formulate the multi-task schedulability problem, prove its NP-Hardness, and propose an approximate algorithm with analysis on the performance bound and complicity. We further extend the proposed algorithm by explicitly altering duty cycles of certain sensor nodes so as to fully support applications with stringent delay requirements to accomplish all tasks. We then design a practical scheduling protocol based on proposed algorithms. We conduct extensive trace-driven simulations to validate the effectiveness and efficiency of our approach with various settings.

Index Terms—Wireless sensor networks, duty-cycling, data forwarding, schedulability

1 Introduction

As an emerging and promising technique, wireless sensor networks (WSNs) have spawned a variety of critical applications in practice, such as environmental monitoring, data collection, object tracking [1], [2], [3], [4], [5], [6], [7], etc. In most applications, sensor nodes are powered by batteries, while frequent replacements of such power sources are normally prohibited [8]. To close the gap between limited energy supplies of sensor nodes and the long-term deployment requirement, recent research works suggest to operate sensor nodes in the *duty-cycling* work mode, where radio chips are not operating all the time. Instead, they are controlled to alternate between the *active* state and the *dormant* state. If the active state of each sensor node accounts for only a small portion in the time domain, e.g., ≤ 5 percent, such networks are referred to as *low-duty-cycling* WSNs. As reported by recent studies [8], [9], [10], [11], idle listening of radio is the major source of battery drain in WSNs. The duty-cycling technique thus significantly reduces the energy consumption of sensor nodes, and as a result, dramatically prolongs the network lifetime.

Although the duty-cycling technique notably increases the network lifetime, excessive challenges are introduced to the packet forwarding efficiency and latency in the network. First, in duty-cycling networks, sensor nodes are not always active. To forward one data packet, the sender needs to hold the packet and wait until the receiver becomes active, which leads to a relatively long per-hop transmission delay. Such a unique per-hop delay in duty-cycling WSNs is known as *sleep latency*. Due to this latency, sensor nodes may not get adequate bandwidth to transmit packets in time, and a data packet may suffer an extremely long delay during the delivery from the source node to the destination node. Second, due to the duty-cycling property, multiple sensor nodes may simultaneously transmit packets to the same receiver during its brief active time, which may incur heavy packet collisions, leading to a further increased per-hop latency. The problem can become even worse if 1) multiple data forwarding tasks with different source-destination pairs coexist in the network and each of those tasks has its own time constraint, and 2) interference hinders concurrent transmissions of nearby wireless links.

There have been many attempts made for scheduling packet forwarding to minimize the transmission delay in duty-cycling WSNs. Most of existing works, however, take relatively simple assumptions on network settings, e.g., routing path for each forwarding task is given in advance or the interference between links are omitted when the network duty cycles are extremely low [2], [12], [13]. None of them tackles the scheduling problem with a general network setting, i.e., considering the limitations and requirements in practical routing choices, interferences, etc. It has been assumed that all transmission tasks can be scheduled within their own deadlines in [12], while it is not true in general duty-cycling WSNs. In particular, given a set of source-destination pairs without specified routing paths, due to duty cycle and interference limitations, not all required data forwarding tasks are guaranteed to be scheduled within their respective delay constraints. Naturally, we are interested in the questions *what is the maximum number of*

- *M. Li and Z. Li are with the School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore. E-mail: {limo, lzjiang}@ntu.edu.sg.*
- *L. Shangguan is with the Department of Computer Science & Engineering, Hong Kong University of Science and Technology, Hong Kong. E-mail: lshangguan@cse.ust.hk.*
- *S. Tang and X.-Y. Li are with the Department of Computer Science, Illinois Institute of Technology. E-mail: {stang7, xli}@cs.iit.edu.*

Manuscript received 15 Sept. 2012; revised 11 Feb. 2013; accepted 14 Feb. 2013. Date of publication 6 Mar. 2013; date of current version 13 Aug. 2014. Recommended for acceptance by X. Cheng.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.65

tasks that could be completed within their deadlines and *how to work out such a schedule*. These two questions essentially query the fundamental capability of duty-cycling WSNs to support real-time applications, which we call **schedulability**. So far as we know, the raised questions are not well understood yet.

In this paper, we investigate the multi-task schedulability problem in duty-cycling WSNs thoroughly and make the following contributions. We formally define the multi-task schedulability problem, prove its NP-Hardness, and propose an approximation algorithm. In particular, given a set of data forwarding tasks, the proposed algorithm outputs a schedule to approximately maximize the number of tasks that can be completed within their time constraints. We analyze the performance bound and time complexity of the proposed algorithm as well. In practice, some applications have stringent requirements to accomplish all data forwarding tasks within their deadlines. To this end, we examine how our algorithm can be extended to support those applications by enforcing only a small number of sensor nodes to increase their duty cycles. We further introduce multi-task data forwarding protocol (MTDF) for practically operating networks to meet application needs. We conduct extensive trace-driven simulations to verify the effectiveness and efficiency of the proposed approaches. The experiments demonstrate an urgent need to develop efficient schedules for multi-task data forwarding. According to simulation results, the system performance is notably improved by our solutions. To the best of our knowledge, this is a pioneer work that explores the schedulability problem in duty-cycling WSNs.

The rest of this paper is organized as follows: related works are reviewed in Section 2. In Section 3, we formally investigate the schedulability problem, and prove its NP-Hardness. In Section 4, we present the approximate algorithm and in Section 5 we extend the algorithm to adjust sensor duty cycles. In Section 6, we evaluate our approach with extensive simulations. We conclude in Section 7.

2 Related Work

There exist a number of studies for scheduling data forwarding tasks in WSNs. In [14], the authors propose a collision-free data aggregation protocol for distributed sensor networks. Authors in [15] introduce a distributed algorithm to calculate the schedule represented by a series of time slots. In [16], the authors design an opportunistic scheduling approach with delay constraints so as to maximize the system throughput. For high data rate wireless sensor networks, [17] proposes a novel scheduling technique named Dynamic Conflict-free schedule.

In the duty-cycling network context, authors in [13] propose an opportunistic flooding protocol for information dissemination. [2] introduces dynamic switch-based forwarding (DSF). DSF achieves the optimized schedule by using the dynamic programming technique. In a most relevant work with this paper [12], the authors study how to schedule multiple tasks such that the work load can be balanced over the sensor network. The authors in [18] propose a practical opportunistic routing scheme. In [12], authors assume that the routing path for each task is given in advance and the network has sufficient bandwidth to simultaneously support all the tasks. Most above existing works simply omit interference between any two different links by assuming such interference is negligible when the duty cycle of sensors is extremely low. This work significantly differs from aforementioned literatures as follows: given a set of source-destination pairs without predetermined routing paths, due to duty cycle and interference limitations, not all required data forwarding tasks are guaranteed to be scheduled within their respect delay constraints. Therefore, this paper investigates a more fundamental and general scheduling problem in duty-cycling WSNs. So far as we know, such a study is still lacking in the community.

There are also a variety of works focusing on the duty cycle adjustment in duty-cycling WSNs. In [19], the authors introduce different approaches for real-time communication in WSNs. They combine the duty cycle adjustment at individual node and the placement of sink nodes to achieve a hybrid system design. In [20], Dutycon is proposed to achieve a dynamic duty cycle control for end-to-end (E2E) delay guarantees in wireless sensor networks. In [21], the authors investigate how to bound communication delay in energy harvesting sensor networks. Unlike this paper, most of them, however, focus on the duty cycle adjustment for one-to-one or one-to-many data delivery in the network.

TABLE 1
Notations in the Problem Formulation

t	Time index
T	Working period
u, v	Indices of sensor nodes
P_u	A set of slots within T for node u to open radio
n	Total number of tasks in the network
c_i	Delay constraint for task i
$I_{e_m}^{e_n}$	Binary indicator which indicates whether links e_m and e_n interfered with each other
$x_{u,v}^{i}(t)$	Binary decision variable indicating whether u transmits the data of $task_i$ to v at time t

3 Problem Formulation

In this section, we present the system model and mathematical formulation of the schedulability problem. We then analyze the hardness of the formulated problem. To facilitate our discussion, key notations used in this section are tabulated in Table 1.

3.1 System Model

In this paper, a wireless sensor network is modeled as an undirected graph $G = (V, E)$, where V and E represent the sets of all sensor nodes and wireless links, respectively. In duty-cycling WSNs, for the sensing purpose, sensor nodes normally work on a periodical operation basis, i.e., the time line of each sensor node is divided into working periods with duration T (e.g., T can be any common multiple of durations of all sensor nodes). In one working period, T is further divided into multiple time slots with equal length.

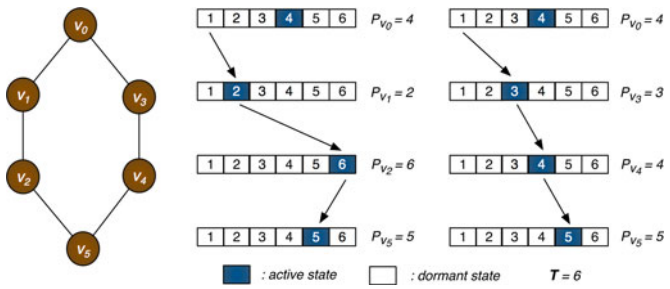


Fig. 1. Illustration of a data forwarding task.

For any sensor node u, it randomly selects several time slots, recorded by a set P_u, in which u activates radio to receive data. In the remaining time slots of T, sensor node u's radio is dormant unless it needs to send data. The random active-state selection policy reduces the chance of interference or collision due to concurrent transmissions from multiple sensor nodes. After determining an active/dormant pattern in one working period, sensor node u will repeat such a pattern over its lifetime [2], [12], [13]. For simplicity of presentation, the length of one time slot is set to 1, which is the minimum time unit in the system.

Interference between two wireless links is characterized by *interference graph* or *conflict table* [17], [22], which is the set of all link pairs (e_m, e_n), where $e_m, e_n \in E$ and $e_m \neq e_n$, such that the minimum distance between either endpoints of e_m and e_n is smaller than the interference range, i.e., links e_m and e_n are interfered with each other. Particularly, we denote $I_{e_m}^{e_n} = 1$, if e_m and e_n are interfered with each other; otherwise, $I_{e_m}^{e_n} = 0$.

A *task* is formally defined as a data forwarding request from a source node to a destination node without pre-calculated routing path. Consider n tasks in the network, each task $task_i$, $1 \leq i \leq n$, can be represented by a triple (v_{s_i}, v_{d_i}, c_i), in which v_{s_i}, v_{d_i}, and c_i are the *source node*, *destination node*, and *time constraint* of $task_i$, respectively. We use $\mathcal{N}$ to denote the set of all tasks. A *schedule* for n tasks records the time and the forwarding sequence for sensor nodes to transmit data. In particular, $x_{u,v}^i(t)$ in the schedule represents that at time t, sensor node u sends the data packet of $task_i$ to sensor node v. The schedule is feasible if 1) at any time, a sensor node can either send or receive data, but not both; 2) any sensor node cannot receive data from multiple senders simultaneously; 3) concurrent transmissions should not be interfered with each other. Furthermore, $task_i$ is successfully scheduled, referring to as $x_{u,v_{d_i}}^i(t) = 1$, where u is a neighbor of v_{d_i} and $t \leq c_i$.

We assume that sensor nodes are synchronized and each node has a local view of the conflict table and neighbors' working patterns. Fig. 1 illustrates a task $(v_0, v_5, 6)$ in an example network. The data are generated at time 1 and there are two possible routing paths to schedule this task: $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_5$ and $v_0 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5$. Given the active/dormant pattern of each sensor node, only the latter choice can successfully finish the task before its deadline.

3.2 Problem Formulation

The *schedulability* problem studied in this paper can be formally described as follows: given a wireless sensor network with working period T, the active/dormant pattern P_u, $\forall u \in V$, n tasks (v_{s_i}, v_{d_i}, c_i), $i \in \mathcal{N}$, where $|\mathcal{N}| = n$, and the interference table, to derive a schedule so as to maximize the number of tasks that can be successfully scheduled within their deadlines. Formally, denote $x_{u,v}^i(t)$ as an **binary** decision variable for $u, v \in V$ and $1 \leq i \leq n$. The decision variable $x_{u,v}^i(t) = 1$, if sensor node u transmits the data of $task_i$ to sensor node v at time t; otherwise, $x_{u,v}^i(t) = 0$. Therefore, for $1 \leq t \leq c^*$, where $c^* = \max_{1 \leq i \leq n}\{c_i\}$, we have

$$\max \quad \sum_{i=1}^{n} \sum_{t=1}^{c_i} \sum_{u \in N_{v_{d_i}}} x_{u,v_{d_i}}^i(t), \tag{1}$$

$$s.t. \quad \sum_{i=1}^{n} \sum_{u \in N_v} (x_{u,v}^i(t) + x_{v,u}^i(t)) \leq 1, u, v \in V \tag{2}$$

$$x_{u,v}^i(t) = 0, t \mid T + 1 \notin P_v, u, v \in V \tag{3}$$

$$x_{u,v}^i(t) + I_{(u',v')}^{(u,v)} \cdot x_{u',v'}^j(t) \leq 1, (u,v), (u',v') \in E \tag{4}$$

$$\sum_{t=1}^{c_i} \sum_{v \in N_u} x_{u,v}^i(t) = \sum_{t=1}^{c_i} \sum_{u \in N_v} x_{v,u}^i(t), u, v \in \overline{V_i}, i \in \mathcal{N} \tag{5}$$

$$x_{u,v_{d_j}}^i(t) = 0, u \in V, i \neq j, i \in \mathcal{N}, \tag{6}$$

where N_u denotes the neighborhood set of sensor node u and $\overline{V_i}$ represents $V/\{v_{s_i}, v_{d_i}\}$. Eq. (2) ensures that at any time slot t, a sensor node can only send or receive one packet, but not both. Eq. (3) restricts that sensor node u can transmit data to node v only when v is active. Eq. (4) guarantees that transmissions for $task_i$ and $task_j$ are not interfered with each other. Eq. (5) refers to flow balance equations on all intermediate forwarding nodes for each $task_i$. Eq. (6) ensures that any destination node j does not receive the data that are not designated to itself, i.e., v_{d_j}. The objective function in Eq. (1) is to maximize the total number of tasks that can be successfully scheduled.

In Fig. 2, we provide an instance of the schedulability problem with four tasks. The optimal strategy can at the most schedule three tasks: $v_1 \rightarrow v_5(4) \rightarrow v_8(6)$, $v_2 \rightarrow v_6(2) \rightarrow v_9(4)$, and $v_4 \rightarrow v_7(4) \rightarrow v_9(6)$, where the number in the bracket indicates the time when the corresponding node receives the data. Different strategies result in a various maximum number of tasks that can be accomplished successfully. For example, if we apply the greedy strategy to schedule as many tasks as possible at time 2, or we apply the greedy strategy to schedule the task with the smallest delay constraint first at time 6, the maximum number of tasks can be completed is only 2. As a matter of fact, the schedulability problem is generally hard and we will examine its hardness in the next section.

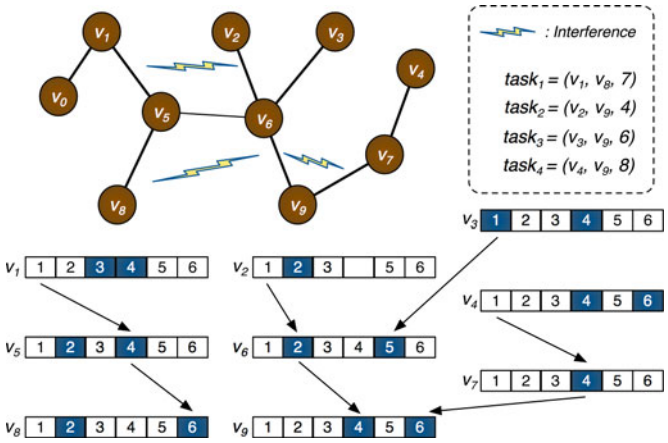


Fig. 2. An instance of the schedulability problem.

3.3 Hardness of the Schedulability Problem

To understand the hardness of the schedulability problem, we define the decision schedulability (DS) problem as: *given an integer $k \leq n$, does the optimal solution of the original schedulability problem equal to k*? We will prove that DS problem is NP-Complete.

Lemma 1. *The DS Problem is NP-Complete.*

The detailed proof is given in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.65. Thus, we have

Theorem 1. *The schedulability problem is NP-Hard.*

4 Protocol Design for Schedulability

Since the schedulability problem is NP-hard, we introduce a heuristic algorithm called Heuristic Algorithm for Schedulability (*HAS*) in this section. The basic idea of *HAS* is to jointly consider both the time urgency of tasks and the interference between wireless links, such that urgent tasks through wireless links not interfered are preferred to be forwarded. Based on our proposed algorithm, we further design a multi-task data forwarding protocol for duty-cycling WSNs in the next section.

4.1 HAS Algorithm Design

The scheduling result for any sensor node v is an $1 \times c^*$ vector, in which the entry in column k can be expressed by a triple $\langle a, i, u \rangle_k$, where i and u indicate task and neighbor indices respectively, and $a \in \{-1, 0, +1\}$. If $a = 0$, i and u are marked as NIL, and sensor node v takes no action at time k. Sensor node v sends the data of $task_i$ to neighbor u, if $a = -1$; otherwise, it receives the data of $task_i$ from neighbor u. The objective of our algorithm is to properly set entries of the scheduling vector for each sensor node, such that the total number of tasks that are successfully scheduled can be approximately maximized.

The detail of the proposed HAS algorithm is given in Algorithm 1. The scheduling process is generally controlled by three sets: $\mathcal{S}_{all}$, $\mathcal{E}_{edg}$ and $\mathcal{T}_{snd}$. More notations used for the protocol design are summarized in Table 2:

TABLE 2
Notations in the Algorithm Design

$\mathcal{S}_{all}$	All sensor nodes having data to forward
$\mathcal{E}_{edg}$	All potential outgoing links from $\mathcal{S}_{all}$
$\mathcal{T}_{snd}$	Nodes in $\mathcal{S}_{all}$ allowed to send data
$\mathcal{P}_{psnd}$	Nodes to be transmitted with higher priority
$\mathcal{L}_{pedg}$	Transmissions after the augmenting operation
$w_{(\langle v,i \rangle,u)}$	Weight for the transmission from v to u for $task_i$
$I_{(\langle v,i \rangle,u)}$	All wireless links interfered with link $(\langle v,i \rangle,u)$
λ	System parameter in Eq. (7)

- At the beginning of a time slot, $\mathcal{S}_{all}$ contains all the sensor nodes that have data to forward. More precisely, an element in $\mathcal{S}_{all}$ can be represented as $\langle v, i \rangle$, indicating sensor node v holding the data packet of $task_i$ to send. Initially, $\mathcal{S}_{all}$ contains $\langle v_{s_i}, i \rangle$, $1 \leq i \leq n$.
- For each $\langle v, i \rangle$ in $\mathcal{S}_{all}$, edges between node v and all its neighbors with smaller hop counts to v_{d_i} compared with v will be included in $\mathcal{E}_{edg}$. In particular, an element in $\mathcal{E}_{edg}$ can be expressed as $(\langle v, i \rangle, u)$, where u is the neighbor of v. Note that $\mathcal{E}_{edg}$ contains all potential outgoing links from $\mathcal{S}_{all}$ in the current time slot.
- $\mathcal{T}_{snd}$ will contain certain sensor nodes in $\mathcal{S}_{all}$ that are allowed to send data in the current time slot. How to make the scheduling decision will be introduced next.

Algorithm 1: The *HAS* Algorithm

Input : $G = (V, E)$ and n tasks $task_i$, $1 \leq i \leq n$.
Output: The scheduling vectors.

```
1 S_all = ⋃_{1≤i≤n} {⟨v_{s_i}, i⟩}; t ← 1;
2 E_edg, T_snd, P_psnd, L_pedg ← NULL;
3 Add a virtual terminal v_d in G and connect v_d to each
  destination node;
4 Label the hop count of each node by BFS starting from
  v_d;
5 while t ≤ c* and S_all ≠ NULL do
6     if is_Aug is true then
7         DutyCycleAdjust(P_psnd, L_pedg, t);
8     DataForwardSchedule(P_psnd, L_pedg);
9     t = t + 1;
```

As aforementioned, the HAS algorithm considers both the time urgency of tasks and the interference influence of wireless links during scheduling. Particularly, at the beginning of time slot t, the time urgency of $task_i$ can be characterized by $c_i - t$, which measures the available time left before the deadline. On the other hand, if sender u sends the data packet of $task_i$ via link $(\langle v, i \rangle, u)$, the corresponding interference influence can be captured by $|I_{(\langle v,i \rangle,u)}|$, where $I_{(\langle v,i \rangle,u)}$ is the set including all wireless links interfered with link $(\langle v, i \rangle, u)$ and $|I_{(\langle v,i \rangle,u)}|$ represents the size of $I_{(\langle v,i \rangle,u)}$. Thus, for each element $(\langle v, i \rangle, u)$ in $\mathcal{E}_{edg}$, we can define its weight as follows:

$$w_{(\langle v,i \rangle,u)} = \lambda(c_i - t) + (1 - \lambda)|I_{(\langle v,i \rangle,u)}|, \tag{7}$$

where $\lambda \in [0,1]$. Although we combine the time urgency and the interference influence through a linear relationship in Eq. (7), such a simple linear relationship is later shown to be quite effective in Section 6. To determine which sensor nodes are allowed to transmit in the current time slot, all edges in $\mathcal{E}_{edg}$ are placed in a priority queue following an increasing order based on Eq. (7). We repeat following operations until the priority queue becomes empty:

- Pop out the first edge (e.g. $(\langle v,i\rangle,u)$) in the priority queue.
- Delete element $\langle v,i\rangle$ from $\mathcal{S}_{all}$ and record it in $\mathcal{T}_{snd}$.
- Delete all the edges that are interfered with $(\langle v,i\rangle,u)$ from both $\mathcal{E}_{edg}$ and the priority queue.

After the priority queue becomes empty, each element $\langle v,i\rangle$ included in $\mathcal{T}_{snd}$ can send data in the current time slot, and the desired link for transmission is recorded in $\mathcal{E}_{edg}$. Moreover, elements left in $\mathcal{S}_{all}$ fail to be scheduled at the current time. The detailed interpretation can be found in Algorithm 2, which serves as a sub-route of Algorithm 1. Based on $\mathcal{T}_{snd}$, $\mathcal{E}_{edg}$ and $\mathcal{S}_{all}$, the scheduling vector of each node for the current time can be updated accordingly.

Algorithm 2: *DataForwardSchedule*$(\mathscr{P}_{psnd}, \mathscr{L}_{pedg})$

```
1  S_all ← S_all − P_psnd, T_snd ← NULL;
2  foreach ⟨v,i⟩ in S_all do
3      foreach neighbor u of v with a smaller hop count do
4          if u is active at time t then
5              add edge (⟨v,i⟩,u) into E_edg;
6  delete edges interfered with links in L_pedg from E_edg;
7  foreach edge (⟨v,i⟩,u) in E_edg do
8      calculate w_(⟨v,i⟩,u) = λ(c_i − t) + (1 − λ)|I_(⟨v,i⟩,u)|;
9      push edge (⟨v,i⟩,u) into a priority queue Q;
10 while priority queue Q is not empty do
11     pop out the first edge ((⟨v,i⟩,u)) from Q;
12     if node u is the destination node v_{d_i} for task_i then
13         delete ⟨v,i⟩ from S_all;
       else
14         delete ⟨v,i⟩ from S_all and record it in T_snd;
15     delete all the edges interfered with (⟨v,i⟩,u) from
       both E_edg and Q;
16 S_all ← S_all ∪ T_snd;
17 update scheduling vectors based on T_snd, E_edg, and S_all;
```

Note that lines 6 and 7 in Algorithm 1 are reserved for the duty-cycling adjustment operation which we will introduce in the next section. At the current stage, line 6 and line 7 in Algorithm 1 will not execute, and both $\mathcal{P}_{psnd}$ and $\mathcal{L}_{pedg}$ equal to $NULL$. We will introduce them in Section 5.

4.2 Performance Lower Bound of HAS

The rationale behind *HAS* is to extend the original graph $G=(V,E)$ in the time domain to implicitly form another graph $G'=(V',E')$, and the scheduling result can be considered as the maximum number of non-interfered paths explored in G'. More precisely, G' can be organized into c^* columns and each column corresponds to one time slot. In particular, any column t contains all the nodes in $\mathcal{S}_{all}$ at the beginning of Algorithm 2's tth loop in Algorithm 1. Edges in E' crossing columns t and $t+1$ include all elements in $\mathcal{E}_{edg}$ at the beginning of Algorithm 2's tth loop. Initially, the first column only contains n original source nodes. Moreover, if a destination v_d appears in column t and there are k sources designated to v_d (not exceed their time constraints yet until t), we artificially place v_d in the column $t+1$ as well and add k non-interfered links between these two v_ds. By doing so, a successful schedule result for an arbitrary $task_i$ can be represented by a non-interfered path in G', starting from v_{s_i} in the first column and ending at v_{d_i} in a c_ith column. Clearly, the length of one path explored in G' can be measured by the time constraint of the task, which will facilitate our later analysis.

In *HAS*, if we only consider the time urgency of tasks, i.e., $\lambda=1$ in Eq. (7), the algorithm is essentially equivalent to select the maximum number of non-interfered paths with least time constraints or path lengths in $G'=(V',E')$. Although, as we will see in Section 6, such an extreme setting fails to provide the best scheduling result, it offers us a performance lower bound for executing Algorithm 1 with the proper setting in general. How to configure λ is postponed to Section 6. In the rest of this section, we exploit the performance lower bound of *HAS*.

We introduce the concepts of *heavy weight* and *light weight* paths in $G'=(V',E')$. A heavy weight path is defined as the one whose time constraint is larger than l, and all remaining paths are light weight paths. l will be specified and adjusted later in Lemma 4. For simplicity of presentation, we denote non-interfered paths returned by the optimal strategy and our proposal as OPT and HAS, respectively. We define that:

- I is the largest number of interfered links in network.
- d is as the maximum node degree in the network.
- m is the number of destination nodes, where $m \le n$.

Lemma 2. *OPT contains no more than $|E'|/l$ heavy paths.*

Proof. We prove it by contradiction. Non-interfered paths in *OPT* are at least edge-disjoint paths. If *OPT* contains more than $|E'|/l$ heavy paths, the total number of edges involved will be greater than $|E'|$, which is a contradiction. □

Lemma 3. *OPT contains no more than $|HAS| \times l \times I$ light weight paths.*

Proof. Denote p_{HAS} as the path interferes one fixed light weight path p_{OPT} in *OPT* for the first time. Thus, all of previously selected edges in *HAS* do not interfere with p_{OPT}, which indicates that both p_{HAS} and p_{OPT} can be selected by our algorithm as well. Since *HAS* chooses p_{HAS}, which implies that the time constraint of p_{HAS} is smaller than that of p_{OPT}. Thus, the weight of p_{HAS} is less than l since p_{OPT} is a light weight path, implying p_{HAS} is a light weight path as well. On the other hand, each edge of one light weight path selected in *HAS* interferes with I different paths in *OPT* in the worst case. Thus, *OPT* contains no more than $|HAS| \times l \times I$ light weight paths. □

Lemma 4. *The approximation ratio of the HAS algorithm with $\lambda=1$ is $\Theta(\sqrt{|E'|})$.*

Proof. Since a path must be either a heavy weight or light weight path, based on Lemmas 2 and 3, we have

$$|OPT| \leq |E'|/l + |HAS| \times l \times I. \quad (8)$$

By setting $l = \sqrt{|E'|}$, we can rephrase Eq. (8) as follows:

$$|OPT| \leq |E'|/\sqrt{|E'|} + |HAS| \times \sqrt{|E'|} \times I$$
$$\Rightarrow |OPT| \leq 2 \times I \times \sqrt{|E'|} \times |HAS|.$$

The last inequality holds as $|HAS| \geq 1$ and $I \geq 1$. □

Lemma 5. *$|E'|$ is no larger than $O(n[(1+d)^{c^*-1} + c^*m])$.*

Proof. As aforementioned, the first column in G' contains n original source nodes. Thus, the number of nodes in the second column is no more than $n + n \times d$. Then, the number of edges crossing the first and the second column is no more than $n(1+d)$. In general, the number of edges crossing column t and column $t+1$ is less than or equal to $n(1+d)^t$. On the other hand, due to the destination nodes, the number of extra edges added between column t and $t+1$ is less than $m \times n$. Therefore, the total number of edges in $|E'|$ is $O(n((1+d)^{c^*} - 1)/d + c^*mn) \simeq O(n[(1+d)^{c^*-1} + c^*m])$. □

Lemma 5 shows that $|E'|$ is only determined by the number of source nodes and destination nodes, time constraints and network density, which is independent with $|E|$ in the original graph. As we will show that *HAS* performs much better when a proper λ is selected instead of simply setting $\lambda = 1$. Based on Lemmas 4 and 5, we already have a performance low bound:

Theorem 2. The number of tasks scheduled by HAS is $\Omega(\sqrt{n[(1+d)^{c^*-1} + c^*m]}))$ approximated compared to $|OPT|$.

5 Duty Cycle Adjustment

In previous two sections, we have studied the schedulability problem in duty-cycling WSNs and proposed the *HAS* problem to approximately maximize the total number of tasks that can be scheduled successfully. However, in many networks, especially in low-duty-cycling WSNs, sensor nodes may not acquire adequate bandwidth to forward packets in their short active time. As a direct consequence, only a small number of tasks can be successfully managed within deadlines even with the optimal scheduling policy, which may fail to meet the stringent application requirements. Representative applications with stringent data delivery requirements include the critical-mission data collection [3], multi-objective real-time tracking [23], voice over senor networks [24], network-wide localization [25], [26], and so on. Tailored for supporting such a type of applications, in this section, we discuss how we can slightly augment the duty cycles of certain sensor nodes, such that most tasks can be finished within their time constraints.

How to optimally perform duty-cycle augmentation, ensuring that the total number of augmented time slots can be minimized, meanwhile all the tasks can be finished within their deadlines, can be proved to be NP-hard. In this section, we demonstrate that how we can extend our *HAS* algorithm to approximately maximize the number of tasks that can be scheduled in time via augmenting duty cycles of certain sensor nodes. Since the duty-cycle augmenting operation increases the energy consumption of the network and the costs of communications and computations, we provide a switching variable is_Aug to control whether the duty-cycle augmenting operation is enabled or not in Algorithm 1. If is_Aug is true, the augmenting operation will be performed. The detail of the duty-cycle augmenting operation is given in Algorithm 3, in which we introduce two new sets: $\mathcal{P}_{psnd}$ and $\mathcal{L}_{pedg}$:

- $\mathcal{P}_{psnd}$ is a sub-set of $\mathcal{S}_{all}$. Compared with those elements in $\mathcal{S}_{all}$ but not in $\mathcal{P}_{psnd}$, items $\langle v, i\rangle$ in $\mathcal{P}_{psnd}$ should be transmitted in the current time slot, while there is no receiver active. Therefore, they will be scheduled via the duty-cycle augmenting operation. How to determine $\mathcal{P}_{psnd}$ will be introduced soon.
- Each element $(\langle v, i\rangle, u)$ in $\mathcal{L}_{pedg}$ indicates that node v will send the data of $task_i$ to node u after the augmenting operation, where $\langle v, i\rangle \in \mathcal{P}_{psnd}$.

Algorithm 3: *DutyCycleAdjust*$(\mathcal{P}_{psnd}, \mathcal{L}_{pedg}, t)$

```
1 P_psnd ⟵ NULL; L_pedg ⟵ NULL;
2 foreach element ⟨v,i⟩ in S_all do
3     if [(c_i − t) − (level_v − 1)] < σ and v is not active
      then
4         select (⟨v,i⟩,u) with the smallest |I_(⟨v,i⟩,u)|, where
          u is v's neighbor;
5         P_psnd = P_psnd ∪ {⟨v,i⟩};
6         L_pedg = L_pedg ∪ {(⟨v,i⟩,u)};
7 foreach (⟨v,i⟩,u) in L_pedg do
8     activate the radio of node u at t;
9     update the t-th entry in the scheduling vector of u
      as ⟨+1,i,u,1⟩_t;
```

To determine which element $\langle v, i\rangle$ in $\mathcal{S}_{all}$ needs to be scheduled through the duty-cycle augmenting operation, we consider two aspects as follows: 1) the time urgency, i.e., $c_i - t$, which measures the available time left before the deadline of the task; and 2) how many time slots are needed *at least* for the data packet of the task to be received by the destination, i.e., $level_v - 1$, where $level_v$ is the hop count of node v. Similar to Eq. (7), we define

$$\overline{w_{\langle v,i\rangle}} = (c_i - t) - (level_v - 1). \quad (9)$$

It is easy to verify that first, as time goes by, $\overline{w_{\langle v,i\rangle}}$ becomes smaller; second, when $\overline{w_{\langle v,i\rangle}}$ is small, $task_i$ is more unlikely to be accomplished in time. Therefore, when $\overline{w_{\langle v,i\rangle}}$ becomes sufficiently small (e.g., smaller than a threshold σ), even all neighbors of node v are dormant, we need to artificially augment one neighbor such that the data of $task_i$ can be sent out immediately.

In Section 4, entries of each scheduling vector are expressed by $\langle a, i, u\rangle_k$. To support the duty-cycle augmenting operation, entries in a scheduling vector will be extended to an quadruple $\langle a, i, u, b\rangle_k$, where the meaning of a, i, u, k is the same as before and $b \in \{1, 0\}$. If $b = 1$, sensor node v will be enforced to be active at time k; otherwise, v follows its original active/dormant pattern.

The detailed algorithm execution can be found in Algorithm 3. Note that, if the duty-cycle operating is disabled, both $\mathcal{P}_{psnd}$ and $\mathcal{L}_{pedg}$ are $NULL$, which do not disturb the execution of the original *HAS* algorithm.

5.1 Protocol Design and Description

Based on the proposed algorithms, including Algorithms 1 to 3 in the *HAS* algorithm, we design a multi-task data forwarding protocol for duty-cycling wireless sensor networks in this section. The MTDF protocol consists of three phases: *scheduling generation, scheduling dissemination* and *node working*.

The *schedule generation* phase generates a scheduling vector for each sensor node v, in which the kth entry $\langle a, i, u, b\rangle_k$ indicates the time when node v should send (receive) the data of $task_i$ to (from) node u. An organizer node (e.g., the sink) is required to execute the *HAS* algorithm to obtain the desired scheduling result. If the organizer is aware of all the information of n tasks initially, it simply disseminates the generated scheduling vector to each corresponding node in the *scheduling dissemination* phase; otherwise, the information of tasks needs to be sent to the organizer before the algorithm execution. Note that a node only needs to receive its own scheduling vector rather than the scheduling vectors of all predecessors. In addition, the scheduling dissemination phase does not cover the nodes that are involved in none of n tasks. Therefore, the communication cost can be largely restricted. When all the sensor nodes involved in the data forwarding of n tasks obtain their scheduling vectors, they start to forward data according to the harvested schedules. The behavior of a sensor node u can be described as follows: at the beginning of each time slot t, node v checks whether the last item b $\langle a, i, u, b\rangle_t$ is 1. If so, node v activates its radio; otherwise, it follows its original active/dormant pattern. Once the radio is activated at time t and the first item a in $\langle a, i, u, b\rangle_t$ is -1, node v sends the data of $task_i$ to node u. If the transmission is successful, node v deletes the data from its buffer. In addition, if the first item a in $\langle a, i, u, b\rangle_t$ is $+1$, node v receives the data of $task_i$ from node u; otherwise, node v keeps dormant.

5.2 Protocol Practical Issues

To make the proposed the *MTDF* protocol available in a variety of practical applications, we still need to address following two practical issues:

Local time synchronization: In Sections 4 and 5, we assume that sensor nodes in the network are synchronized. As a matter of fact, it is sufficient for sensor nodes to know the active/dormant patterns of its neighbors, thus the global synchronization is not a compulsory. According to [27], an accuracy of $2.24\ \mu$s local synchronization can be achieved by simple and low-cost techniques with the cost of exchange several bytes among neighbors every 15 minutes. Since one time slot is typically longer than $10{,}000\ \mu$s [28], the achieved $2.24\ \mu$s accuracy is far more than sufficiency. In addition, a transmission in the network is not required to start at the beginning of an active slot, this further relaxes the requirement for accuracy of time synchronization. If initial clock offset is corrected, nodes can also be synchronized via [29].

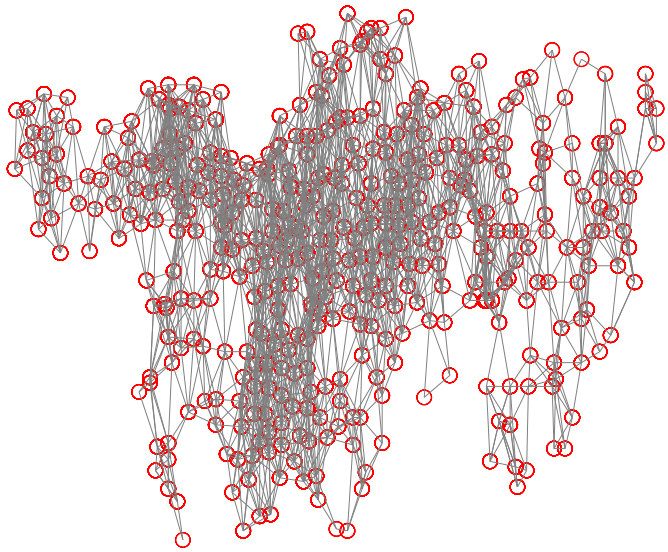


Fig. 3. Harvested networking topology from GreenOrbs.

Unreliable wireless links: For a typical TinyOS packet with a normal packet size, e.g., 50 bytes, a sensor node with $CC2420$ radio can transmit the same packet more than 10 times within *one* active time slot. It implies that, even though wireless links are usually unreliable, as long as the link quality between two nodes are not extremely low (e.g., larger than 30 percent), at least 99 percent of packets can be successfully transmitted within an active time slot. As existing link estimation protocols can filter links with low link qualities [2], [30], the requirement of at least 30 percent link quality is reasonable in current WSNs.

6 Performance Evaluation

In this section, we evaluate the scheduling performance of MTDF in comparison with the recent proposed SAG[1] [12] and the best effort algorithm (BEA). Given predetermined routing paths for each task, SAG derives scheduling results by a load balancing technique to fully utilize the network bandwidths, and each sensor node in BEA greedily transmits packets to the receiver waking up earlier than other receivers. To test a realistic network setting, the simulations are conducted with a real trace harvested from GreenOrbs [1]. GreenOrbs is a long-term and large-scale wireless sensor network deployed in the forest, which contains 433 nodes and has continuously worked over one year. From the harvested trace over six months, we observe that the dynamics of wireless links result in fluctuating of the network topology. To mimic the link estimation for real data transmissions, we filter out lossy links with small RSSI values. In particular, links with the packet reception ratio (PRR) lower than 30 percent or RSSI smaller than -80 dB are excluded by the filter. By doing so, we obtain a stable network topology for simulations in Fig. 3. The topology includes 433

1. Since SAG requires that routing path for each task should be given in advance, we perform a BFS search and assign the shortest path for each source-destination pair before the algorithm execution.

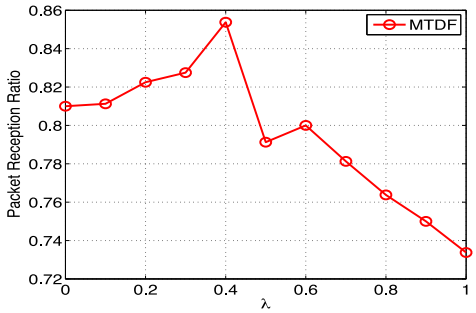


Fig. 4. Packet reception ratio versus λ in Eq. (7).

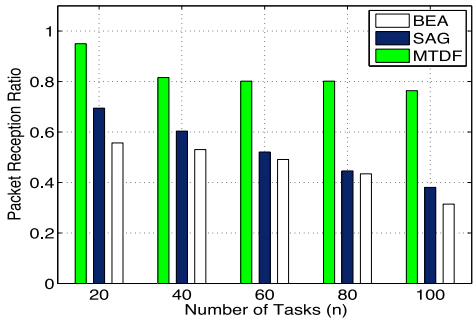


Fig. 5. Packet reception ratio versus number of tasks.

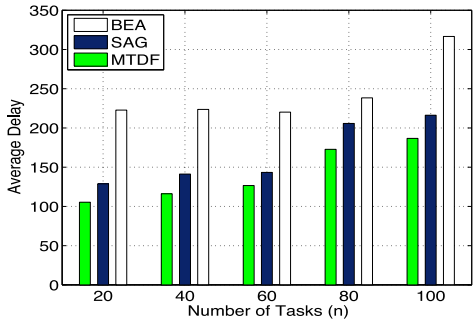


Fig. 6. Average delay versus number of tasks.

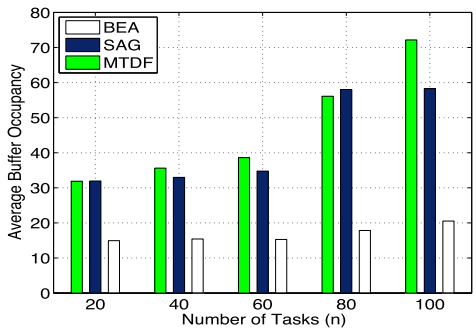


Fig. 7. Buffer occupancy versus number of tasks.

nodes and 6,567 links with relatively good quality. The simulation results show that there is an urgent need to introduce an efficient strategy for multi-task scheduling in duty-cycling WSNs, especially with high data rates and low duty cycles. Our proposed MTDF approach dramatically improves the system performance.

6.1 Experimental Setting

In the trace, sensor nodes are deployed in a $700\,\mathrm{m} \times 200\,\mathrm{m}$ rectangle field with the default transmission

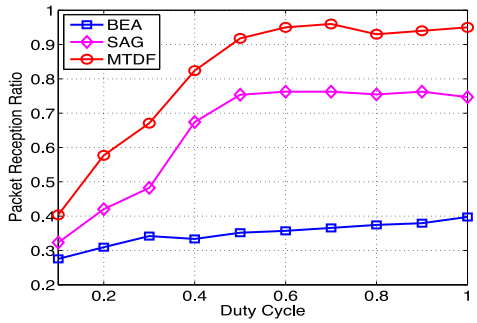


Fig. 8. Packet reception ratio versus duty cycle.

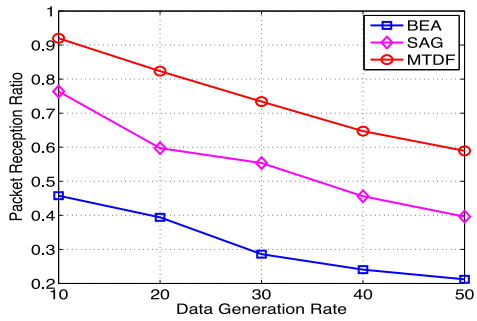


Fig. 9. PRR versus data generation rate.

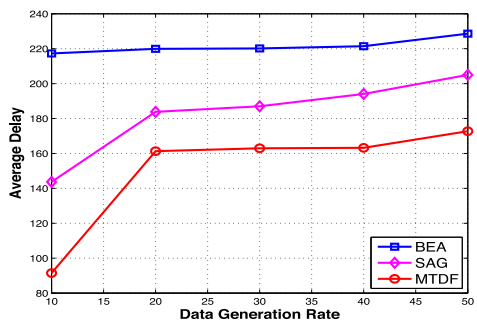


Fig. 10. Average delay versus data generation rate.

power, and the interference range is set as twice as the transmission range [22]. The buffer size is set based on the Telosb mote [31] and link qualities are quantified according to the long-term pair-wise link measurements. Buffer occupancy and link losses are recorded in the simulations. One time slot is set to $10{,}000\,\mu\mathrm{s}$ [21], [28]. We investigate the impact of the duty cycle by varying the duty cycle from 10 to 100 percent and the default duty cycle is 30 percent. In addition, impacts of the number of tasks, delay constraints, data rates and system parameters are also examined. We use *packet reception ratio* to qualify various scheduling algorithms, which is

$$PRR = \frac{\text{No. of pkts received by destinations within deadlines}}{\text{No. of pkts sent by source nodes}}.$$

6.2 Experimental Results

From Figs. 4, 5, 6, 7, 8, 9, 10, and 11, we investigate the multi-task scheduling without duty-cycling augmentation. The duty-cycling augmenting operation is examined by Figs. 12, 13, 14, and 15.

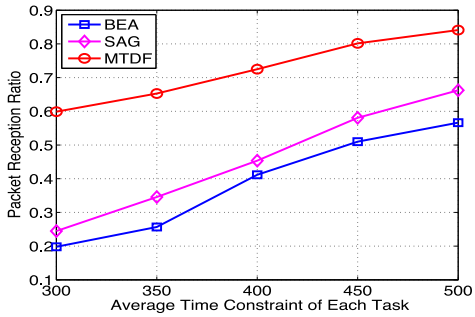


Fig. 11. PRR versus average time constraint of each task.

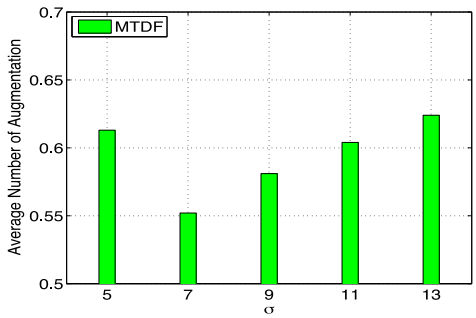


Fig. 12. Average number of augmentation versus σ.

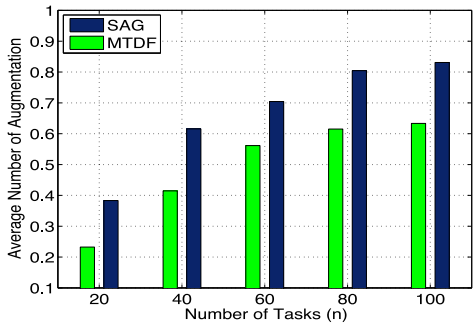


Fig. 13. Number of augmentation versus tasks.

6.2.1 Impact of the System Parameter λ

In Fig. 4, we vary λ from 0 to 1 and depict corresponding *PRR*s. Fig. 4 reveals that if we merely focus on the time urgency of tasks ($\lambda = 1$) or the interference influence of wireless links ($\lambda = 0$) during scheduling, the system performance deteriorates. In particular, when $\lambda = 1$, the performance of MTDF is much worse compared with other proper settings that we have mentioned in Section 4.2 already. From Fig. 4, we can notice that the best performance is achieved with the consideration of both the time urgency and the interference influence, which is much better above the performance lower bound we derive with $\lambda = 1$ in Section 4.

6.2.2 Impact of the Number of Tasks

In this set of simulations, the number of tasks varies from 20 to 100, and each source node has 20 data packets to deliver. The duty cycle is set to be 30 percent and the average time constraint is 450. As the number of tasks increases, we examine a series of performance metrics, illustrated from Figs. 5, 6, and 7.

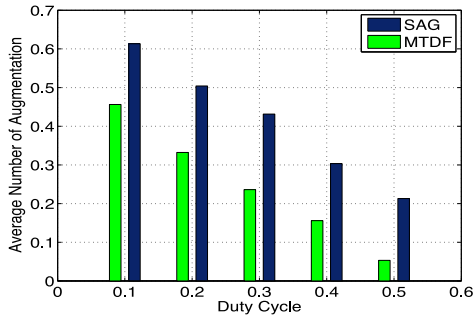


Fig. 14. Number of augmentation versus duty cycle.

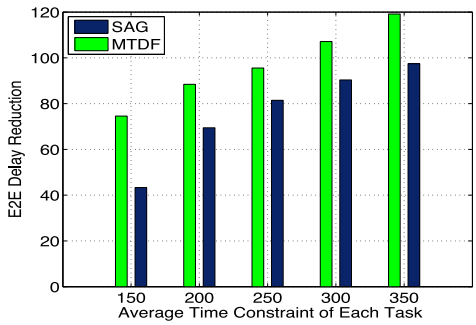


Fig. 15. E2E delay reduction versus deadline.

Fig. 5 depicts how *PRR* is impacted by the number of tasks. With a growing number of tasks that are supported in the network, *PRR* decreases with all three approaches. However, among all three algorithms, MTDF experiences the slowest performance degradation. The *PRR* of MTDF achieves around 0.8 or above in all cases, while *PRR*s of *SAG* and *BEA* drop below 0.4 when $n = 100$. Compared with *SAG* and *BEA*, the performance improvement achieved by MTDF is as high as 50.2 and 58.8 percent, respectively. The rapid performance drop of *SAG* is mainly because *SAG* has subtle different optimization objectives, where *SAG* focuses more on alleviating the routing congestions for load balance. In Fig. 6, we further investigate the average delay of each successfully scheduled task in different systems. Due to limited network resources, the average delay is expected to become longer as the number of tasks increases. From Fig. 6, we observe that both *SAG* and MTDF have much shorter delay than *BEA*, while MTDF outperforms *SAG* with all settings of task numbers. Figs. 5 and 6 jointly demonstrate that MTDF not only schedules the largest number of tasks within desired deadlines, but also achieves the shortest average delay to schedule those tasks.

As the number of tasks becomes larger, the average buffer usage also increases. *BEA* leads to the lowest buffer occupancy as depicted in Fig. 7. When the number of tasks is small in the network, *SAG* and MTDF have similar buffer occupancy performance. Nevertheless, *SAG* might have slightly smaller buffer usage since its routing path is longer than MTDF on average. When the task number is large (i.e., $\geq$80), the buffer usage of MTDF is apparently larger than *SAG* mainly because *SAG* approaches its network capacity earlier than MTDF. The scheduling assignment might get dropped at some intermediate nodes due to buffer overflow. Those missing assignments in *SAG* are not included into the "buffer occupancy" in Fig. 7.

6.2.3 Impact of Duty Cycle

As the duty cycle increases, the bandwidth inside the network will increase as well. As a direct consequence, more tasks have the opportunity to be scheduled before their time constraints. As shown in Fig. 8, compared with BEA, PRRs of both SAG and MTDF grow up rapidly with the increased duty cycle. The improvements achieved by SAG and MTDF are up to 53.3 and 61.7 percent, respectively. Meanwhile, MTDF outperforms SAG 16.4 percent on average. Due to the link loss, the PRR of MTDF fails to achieve 1 even when duty cycle is close to 100 percent. Fig. 8 explicitly suggests the need of developing such an efficient multi-task scheduling algorithm for low-duty-cycling WSNs.

6.2.4 Impact of Data Generation Rate

The data generation rate can be adjusted by changing the number of data packets generated for each task. As data generation rate increases, the available network resources to support multiple tasks become lean. In addition, due to more serious channel contention and interference, RPP is expected to significantly decrease. In Fig. 9, the decreasing of PRR for *BEA*, *SAG*, and MTDF is 53.6, 48.1, 35.9 percent, respectively. MTDF experiences the slowest performance degradation. Fig. 9 further implies that when the data rate is high, the need for an efficiency multi-task scheduling is also urgent.

Fig. 10 reveals that there is no significant delay variance when data generation rate increases in the network. Both *SAG* and MTDF outperform *BEA* in all settings. In addition, MTDF can achieve a shorter average delay compared with *SAG*. According to statistics, the average delay performance improvement of MTDF is 12 percent.

6.2.5 Impact of Average Time Constraint of Each Task

In Fig. 11, we vary the average time constraint of each task from 300 to 500 and illustrate the results. Fig. 11 shows that MTDF achieves the best PRR performance compared with *SAG* and *BEA*, and the performance improvements are up to 59.1 and 66.9 percent, respectively. In addition, we can observe that both *SAG* and *BEA* are more sensitive to the time constraint than MTDF. Statistics show that the PRR reduction of MTDF, *SAG*, and *BEA* is 28.8, 63, and 65 percent, respectively. MTDF experiences the smallest PRR reduction.

6.2.6 Impact of System Parameter σ

From Figs. 12, 13, 14, and 15, we investigate the scheduling performance in low-duty-cycling networks with the duty cycle augmenting approach discussed in Section 5. *Average Number of Augmentation* in Figs. 12, 13, and 14 is defined as the ratio of the extra active time slots in one working period and the original working period T. Essentially, Average Number of Augmentation characterizes the energy cost that the network pays for achieving a high PRR in low-duty-cycling WSNs. In Fig. 12, the duty cycle of each sensor node is set to be 10 percent. There are 40 tasks in the network and the data rate of each source node is 20. From Fig. 8, we know that the PRR of MTDF is around 40 percent in such a network, and in this simulation, we require that at least 95 percent data should be received by destination nodes within their deadlines (we do not require a 100 percent PRR as it may not be feasible due to link loss). Fig. 12 shows that to achieve a high PRR, involved sensor nodes need to augment a substantial number of extra time slots. In this simulation, this number is around 60 percent. Therefore, Fig. 12 implies that the desired PRR in applications should be determined with the consideration of both the importance of the harvested data and the energy consumption of the network. The tradeoff between these two aspects will serve as a promising future work of this paper. On the other hand, the threshold σ for Eq. (9) should be chosen carefully. In the rest of this section, σ is set to be 7, as suggested in Fig. 12.

6.2.7 Impact of Duty Cycle Augmentation

Since the original *SAG* does not consider the duty cycle augmentation, we transplant our duty cycle adjustment module into *SAG* for a fair comparison. On the other hand, as *SAG* cannot achieve an adequately high PRR, we relax the requirement of PRR from 95 to 75 percent in this section. *BEA* is not included in the comparison since its PRR fails to be above 75 percent even with 100 percent duty cycles of sensor nodes.

As the number of tasks increases, more time slots should be activated in the network. Fig. 13 indicates that to achieve the same PRR, MTDF can augment less number of time slots than *SAG*. From statistics, the energy saved by MTDF is up to 31.2 percent compared with *SAG*. On the other hand, if the initial duty cycle of each sensor node becomes larger, the number of time slots to be augmented should become smaller accordingly. As expected, MTDF augments less number of time slots than *SAG* in Fig. 14. On average, MTDF outperforms *SAG* 45.3 percent.

Another advantage of the duty cycle augmentation is to reduce the end-to-end data forwarding delay. As sensors become active more frequently, the one-hop sleep latency will decrease. As a result, the overall E2E delay decreases as well. In Fig. 15, we indeed observe such a phenomena for both MTDF and *SAG*, while MTDF can reduce more E2E delay compared with *SAG*.

7 Conclusion and Future Work

In this paper, we investigate the multi-task schedulability problem in duty-cycling WSNs. We mathematically formulate the schedulability problem, prove its NP-Hardness [32], and propose an approximate algorithm with the analysis on performance bound and complexity. We further extend our algorithm by augmenting duty cycles of certain sensor nodes to support the applications that stringently require to collect approximately all data from the network within deadlines. Proposed algorithms are incorporated in a practical protocol named *MTDF* for operating over actual sensor networks. We conduct extensive trace-driven experiments to evaluate the performance of our proposed protocol and algorithms. In the future, we plan to fully implement MTDF with a prototype system and examine the system performance in practical networking environments. We are particularly interested in addressing the uncertainty of the

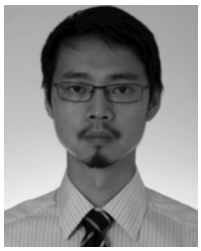
node-connectivity and link-interference information used in the scheduling due to network dynamics.

ACKNOWLEDGMENTS

This study was supported by Singapore MOE AcRF Tier 2 grant MOE2012-T2-1-070. This work was also partially supported by NSFC Grant No. 61272456.

REFERENCES

[1] "GreenOrbs," http://greenorbs.org, 2013.
[2] Y. Gu and T. He, "Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links," *Proc. Fifth Int'l Conf. Embedded Networked Sensor Systems (Sensys)*, 2007.
[3] C. Huang, T. Lin, L. Chen, and P. Huang, "Xd: A Cross-Layer Designed Data Collection Mechanism for Mission-Critical Wsns in Urban Buildings," *Proc. IEEE 10th Int'l Conf. Mobile Data Management: Systems, Services and Middleware (MDM)*, 2009.
[4] Z. Yang and Y. Liu, "Quality of Trilateration: Confidence-Based Iterative Localization," *IEEE Trans. Parallel and Distributed Systems*, vol. 21, no. 5, pp. 631-640, May 2010.
[5] J. Lian, Y. Liu, K. Naik, and L. Chen, "Virtual Surrounding Face Geocasting in Wireless Ad Hoc and Sensor Networks," *IEEE/ACM Trans. Networking*, vol. 17, no. 1, pp. 200-211, Feb. 2009.
[6] Z. Li, Y. Liu, M. Li, J. Wang, and Z. Cao, "Exploiting Ubiquitous Data Collection for Mobile Users in Wireless Sensor Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 24, no. 2, pp. 312-326, Feb. 2013.
[7] Y. Liu, Y. Zhu, L. M. Ni, and G. Xue, "A Reliability-Oriented Transmission Service in Wireless Sensor Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 22, no. 12, pp. 2100-2107, Dec. 2011.
[8] P. Dutta, J. Taneja, J. Jeong, X. Jiang, and D. Culler, "A Building Block Approach to Sensornet Systems," *Proc. Sixth ACM Conf. Embedded Network Sensor Systems (SenSys)*, 2008.
[9] Z. Cao, Y. He, and Y. Liu, "L2: Lazy Forwarding in Low Duty Cycle Wireless Sensor Networks," *Proc. IEEE INFOCOM*, 2012.
[10] T. Zhu, Z. Zhong, Y. Gu, T. He, and Z. Zhang, "Leakage-Aware Energy Synchronization for Wireless Sensor Networks," *Proc. ACM Seventh Int'l Conf. Mobile Systems, Applications, and Services (MobiSys)*, 2009.
[11] J. Chen, W. Xu, S. He, Y. Sun, P. Thulasiraman, and X. Shen, "Utility-Based Asynchronous Flow Control Algorithm for Wireless Sensor Networks," *IEEE J. Selected Areas in Comm*, vol. 28, no. 7, pp. 1116-1126, Sept. 2010.
[12] S. Xiong, J. Li, M. Li, J. Wang, and Y. Liu, "Multiple Task Scheduling for Low-Duty-Cycle Wireless Sensor Networks," *Proc. IEEE INFOCOM*, 2011.
[13] S. Guo, Y. Gu, B. Jiang, and T. He, "Opportunistic Flooding in Low-Duty-Cycle Wireless Sensor Networks with Unreliable Links," *Proc. ACM MobiCom*, 2009.
[14] B. Yu, J. Li, and Y. Li, "Distributed Data Aggregation Scheduling in Wireless Sensor Networks," *Proc. IEEE INFOCOM*, 2009.
[15] A. Rao and I. Stoica, "Adaptive Distributed Time-Slot Based Scheduling for Fairness in Multi-Hop Wireless Networks," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS)*, 2008.
[16] S. Tan, D. Zheng, J. Zhang, and J. Zeidler, "Distributed Opportunistic Scheduling for Ad-Hoc Communications under Delay Constraints," *Proc. IEEE INFOCOM*, 2010.
[17] O. Chipara, C. Lu, and J. Stankovic, "Dynamic Conflict-Free Query Scheduling for Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Network Protocols (ICNP)*, 2006.
[18] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low Power, Low Delay: Opportunistic Routing Meets Duty Cycling," *Proc. ACM/IEEE 11th Int'l Conf. Information Processing in Sensor Networks (IPSN)*, 2012.
[19] Y. Gu, T. He, M. Lin, and J. Xu, "Spatiotemporal Delay Control for Low-Duty-Cycle Sensor Networks," *Proc. IEEE IEEE Real-Time Systems Symp. (RTSS)*, 2009.
[20] X. Wang, X. Wang, G. Xing, and Y. Yao, "Dynamic Duty Cycle Control for End-to-End Delay Guarantees in Wireless Sensor Networks," *Proc. IEEE Int'l Workshop Quality of Service (IWQoS)*, 2010.
[21] Y. Gu, T. He, M. Lin, and J. Xu, "Spatiotemporal Delay Control for Low-Duty-Cycle Sensor Networks," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS)*, 2010.
[22] Y. Ding, Y. Yang, and L. Xiao, "Multi-Path Routing and Rate Allocation for Multi-Source Video On-Demand Streaming in Wireless Mesh Networks," *Proc. IEEE INFOCOM*, 2011.
[23] O. Chipara, C. Lu, and G. Roman, "Real-Time Query Scheduling for Wireless Sensor Networks," *Proc. IEEE Real-Time Systems Symp. (RTSS)*, 2007.
[24] R. Mangharam, A. Rowe, R. Rajkumar, and R. Suzuki, "Voice over Sensor Networks," *Proc. IEEE Real-Time Systems Symp. (RTSS)*, 2006.
[25] Z. Yang and Y. Liu, "Understanding Node Localizability of Wireless Ad-Hoc Networks," *IEEE Trans. Mobile Computing*, vol. 11, no. 8, pp. 1240-1260, Aug. 2012.
[26] L. Ni, Y. Liu, Y. Lau, and A. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," *ACM Wireless Networks*, vol. 10, no. 6, pp. 701-710, 2004.
[27] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The Flooding Time Synchronization Protocol," *Proc. ACM Second Int'l Conf. Embedded Networked Sensor Systems (SenSys)*, 2004.
[28] P. Dutta and D. Culler, "Practical Asynchronous Neighbor Discovery and Rendezvous for Mobile Sensing Applications," *Proc. Sixth ACM Conf. Embedded Network Sensor Systems (SenSys)*, 2008.
[29] Z. Li, W. Chen, C. Li, M. Li, X. Li, and Y. Liu, "Flight: Clock Calibration Using Fluorescent Lighting," *Proc. ACM MobiCom*, 2012.
[30] M. Zamalloa, K. Seada, B. Krishnamachari, and A. Helmy, "Efficient Geographic Routing Over Lossy Links in Wireless Sensor Networks," *ACM Trans. Sensor Networks*, vol. 4, no. 3, article 12, 2008.
[31] "TelosB," http://www2.ece.ohio-state.edu/~bibyk/ee582/telosMote.pdf, 2013.
[32] T.H. Cormen et al., *Introduction to Algorithms,* second ed., MIT Press, 2001.



Mo Li (M'06) received the BS degree from the Department of Computer Science and Technology from Tsinghua University, China, in 2004 and the PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology in 2009. He received ACM Hong Kong Chapter Prof. Francis Chin Research Award in 2009 and Hong Kong ICT Award C Best Innovation and Research Grand Award in 2007. He is currently an assistant professor at the School of Computer Engineering of Nanyang Technological University, Singapore. His research interest includes wireless sensor networking, pervasive computing, mobile and wireless computing. He is a member of the IEEE and ACM.

Zhenjiang Li (M'12) received the BE degree from the Department of Computer Science and Technology, Xi'an Jiaotong University, China, in 2007, the Mphil degree from the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, in 2009, and the PhD degree from the Department of Computer Science and Engineering of Hong Kong University of Science and Technology, in 2012. His research interests include distributed systems, wireless sensor networks, wireless and mobile systems. He is a member of the IEEE and ACM.

Longfei Shangguan received the BE degree from the Department of Software Engineering, Xidian University, China, in 2011. He is currently a third year PhD student of Hong Kong University of Science and Technology. His current research interest is wireless sensor networks. He is a student member of the IEEE.

Shaojie Tang (S'09) received the BS degree in radio engineering from Southeast University, Nanjing, China, in 2006, and is currently working toward the PhD degree in computer science at the Illinois Institute of Technology, Chicago. His research eld is on algorithm design, optimization, security of wireless networks, electronic commerce, as well as online social network. He is a member of the IEEE.

Xiang-Yang Li (SM'08) received the BEng degree in computer science and the bachelor's degree in business management from Tsinghua University, Beijing, China, in 1995, and the MS and PhD degrees in computer science from the University of Illinois at Urbana Champaign in 2000 and 2001, respectively. He has been an associate professor of computer science with the Illinois Institute of Technology, Chicago, since 2006 and was an assistant professor from 2000 to 2006. He is a senior member of the IEEE.