Doctoral Thesis

Methods for Extending Lifetime in Wireless Sensor Networks

by

Qian Zhao

March 2016

Graduate School of Applied Informatics

University of Hyogo

Abstract

This doctoral thesis addresses several issues related to the foremost concerned problem in wireless sensor networks (WSNs): the restricted battery energy. In order to extend the lifetime of wireless sensor network systems, this thesis investigates how to reduce the energy consumption from the main energy consumer in wireless sensor networks: energy consumed in sensors, tasks that manipulate sensors and wireless communications.

Nowadays, wireless sensor nodes are becoming more and more common in various settings and require a long battery life for a better maintainability. However, since most sensor nodes are powered by batteries, energy efficiency is a critical problem affecting our ability to successfully and efficiently maintain WSNs. In many cases, it is extremely difficult or impossible to maintain sensor nodes, for example, when they are deployed in tunnels, oceans, volcanoes, or other dangerous and/or difficult-to-reach places. Moreover, the increasing complexity of WSN systems leads to an increased energy consumption. Due to these reasons, sensor nodes should be able to run on batteries for prolonged periods of time without the need for replacement. Reducing energy consumption in such systems is urgent for further improvement of the WSN systems.

In WSNs, electric power is primarily consumed by wireless communication, sensors in a node, and CPUs where tasks that are used to control sensors. In this doctoral thesis, we investigate problems that cause short battery life in WSNs from two perspectives, and give appropriate methods to solve those problems. First, from the perspective of the execution forms of sensors in a sensor node, we note that simultaneous sensor activation generates high peak power consumption. Therefore, battery voltage drops quickly, and sensors stop working even though some useful charge remains in the battery. Moreover, tasks should have capability to activate or deactivate sensors due to sensors' execution form in an energy efficient manner. Second, from the perspective of wireless communication, communication distances must be considered in minimizing energy consumption. Another problem is the energy hole problem, which is known to cause non-uniform energy drains in many communication topologies, results in premature termination of entire networks. Moreover, since some sensor nodes in a WSN may be unreliable, it must be tolerant to faults. The goal of this doctoral thesis, therefore, is to describe novel algorithms for separately solving these problems in order to extend the lifetime of WSNs. In order to verify the effectiveness of the proposed algorithms, we conduct simulations to evaluate the battery life extension by utilizing these algorithms, and the simulation results showed the algorithms' superiority.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Looking back at the history of human civilizations in thousands of years, there is no period like we are living now: strength global economy, top speed developing technology, refined infrastructure etc., which makes our age the most outstanding in history. Out of question, the industrial revolution[1] is the turning point of this rapid progress. From the first to the third industrial revolution, technology innovation has lead a tremendous change of our world. Now we are on the edge of the third industrial revolution (digital revolution) going to the fourth, in which "smart cyber physical technology" is leading this change. Especially, in the fourth industrial revolution (Industrie 4.0) [1], a concept of "Smart Factory" has been proposed. It is a collective term for technologies and concepts of value chain organization which draws together Cyber Physical Systems (CPS), the Internet of Things (IoT), and the Internet of Services (IoS). Within the modular structured Smart Factories of Industrie 4.0, CPS monitor Physical processes, create a virtual copy of physical world and make decentralized decisions. Over the IoT, CPS communicate and cooperate with each other and humans in real time. Via the IoS, both internal and cross-organizational services are offered and utilized by participants of the value chain [1]. Thanks to the great efforts of smart humans, such kind of systems are becoming pervasive in our earth.

In view of Industrie 4.0, we have no choice but to talk about the Wireless Sensor Network (WSN) System since it is an important component of Industrie 4.0, owing to wireless sensors are indispensable base devices of CPS and IoT. In this doctoral thesis, we will raise some major issues exist in the WSN systems, analyze them in detail and present our new effective resolutions.

[1]Three industrial revolutions have been occurred until now, and the main contribution of the three industrial revolutions are steam power, electric power and digital technology, respectively.

1.1 Motivation

A wireless sensor network (WSN) system is a kind of pervasive computing system that consists a large number of wireless sensor nodes able to collect environmental measurements, such as temperature, sound, light and .etc. These sensing data are finally transmitted to a WSN application that makes decisions based on the sensing data by using a wireless communication device such like ZigBee and Wi-sun. Wireless sensor nodes are now extensively used in many applications to improve people's lives and for security. For example, they are used to manage air conditioners, lights, and alarms, to save electric power, and to warn people of danger by sensing environmental conditions. Sensor nodes are used for infrastructure elements such as bridges and tunnels to detect conditions that could lead to problems such as a collapse, by sensing reflected sound waves. Wireless sensor networks are fast becoming ubiquitous.

However, as most sensor nodes in wireless sensor networks are powered by batteries, the battery life is a problem affecting our ability to maintain them. In many cases it is impossible to maintain sensor nodes, such as when they are deployed in tunnels, oceans, volcanoes, or other dangerous, difficult-to-reach places. Therefore, sensor nodes should be able to run on batteries for prolonged periods without needing to be replaced, the aim being maintenance free operation. In practical, considering the lifetime of digital devices which is about 10 years, a device which can operate on a battery about 10 years can be said as maintenance free. In WSNs, electric power is primarily consumed by wireless communication, sensors in a node, and CPUs where tasks that are used to control sensors. In order to extend the lifetime of WSNs, these energy consumption should be reduced as much as possible. Therefore, reducing energy consumption from these consumers are the main topic of this thesis.

We investigate problems that cause short battery life in WSNs from two perspectives. First, from the perspective of the execution forms of sensors in a sensor node, taking advantage of an experiment of examining sensor schedules, we note that simultaneous sensor activation generates high peak power consumption. High peak power consumption causes battery voltage drop to the operation voltage quickly, hence, sensors stop working even though some useful charge remains in the battery. Therefore, appropriate sensor execution form is required. However, sensors are required to finish their data acquisitions before the determined deadlines, hence, time constrains should be considered in the sensor execution form. Moreover, tasks should have capability to activate or deactivate sensors due to sensors' execution form in an energy efficient manner. Energy efficient task execution form should be considered accompany with the form of sensor execution. Second, from the perspective of wireless communications, energy consumption is mainly affected by communication distance and the size of transmitted data, especially the energy consumption grows in proportion to 2nd$\sim$6th power of the communication distance. Hence, from the viewpoint of topology control, communication distances must be reduced as much as possible by control the network topology. However, another problem is the energy hole problem, which is known as a reason causing non-uniform energy drains in many communication topologies. This is because a node will terminates quicker than the others if it relays data more often than the others, results in premature termination of entire networks. The strategy of avoiding energy hole problem should be taken into consideration in the design of topology control methods. Moreover, since sensor nodes may be damaged in some harsh environment or some sensor nodes run out of energy faster than others due to some other reasons, it is important for a WSN to tolerate such faults. Therefore, a fault tolerance is required when optimizing the communication topology.

1.2 Contributions

The main contributions of this thesis are new algorithms for solving the raised energy efficient problems in WSNs.

From the first perspective, we need a way of emptying the battery before of when the battery voltage reached the operation voltage. We devised three energy-efficient algorithms: an EDF Based Scheduling Algorithm (EDF-BSA), a Uniform Distribution Scheduling Algorithm (UDSA) and a Blank Filling Scheduling Algorithm (BFSA). During the design and simulation of these scheduling algorithms, we found that BFSA is the most efficient solution in that when the battery voltage reaches operation voltage, the battery life is maximized. Therefore, BFSA is capable of efficiently dealing with this problem. Moreover, the tasks should have the capability to activate or deactivate sensors simultaneously when they are to be executed periodically. We devised two task execution scheduling algorithms, DVFS-enabled periodical task execution algorithm (DVFS-PTEA) and execution time scaling periodical task execution algorithm (ETS-PTEA) to solve the problem.

From the second perspective, in order to shorten the communication distance as well as relax the energy hole problem for reducing energy consumption of data communication, we devised an Energy Hole Aware Energy Efficient Communication Routing Algorithm (EHAEC). For further solve the energy hole problem, we proposed two route switching algorithms Tired Node Resting Switching Algorithm (TINORESA) and Complementary Switching Algorithm (COMSA) in order to uniformly distribute the energy consumption of sensor nodes to avoid energy holes. For tolerating node failures, we also proposed two kinds of provisioned tolerance algorithms. EHAEC for one-fault tolerance

(EHAEC-1FT) is proposed to tolerate one node failure in the network, and Active Spare Selecting Algorithm (ASSA) is proposed to designate backups for critical nodes. Moreover, we show that EHAEC-1FT can be extended to X-nFT for any routing algorithm X to tolerate n-1 node failures.

1.3 Outline

The rest of the thesis is mainly divided into two parts. The first part discuss problems from the perspective of sensor and task scheduling, while the second part discuss problems from the perspective of wireless communications. In the first part, Chapter 2 addresses studies related scheduling techniques, Chapter 3 defines the problem, Chapter 4 presents the proposed algorithms, Chapter 5 shows the results of simulations of each of these scheduling algorithms. In the second part, Chapter 6 present related works of energy efficient problems in wireless communications, Chapter 7 and Chapter 8 present the energy efficient routing algorithms and the fault tolerant algorithms, respectively, and Chapter 9 shows the simulation results. Finally, Chapter 10 conclude this work and raises some real applications which our proposed methods adapt.

This doctoral thesis is based on the original works of the author. Chapters 2 to 5 are based on [51, 54], Chapters 6 to 9 are based on [52, 53, 55]. And works [56, 57, 58, 59, 60] are some of the author's publications that related to this thesis.

Part I

Sensor and Task Scheduling

Chapter 2

Related Works

In this chapter, we introduce some concepts and some pieces of research which relate to scheduling and energy-efficient issues. In embedded systems, some of those systems which are required to finish their missions before a specific time constraint are known as embedded real-time systems. In this kind of systems, time constraint is defined as that a control task must produce its result within a specific deadline, which is defined based on the requirement of the system. In addition, activities that require regular activation in a real-time system should be handled as periodic tasks,. In the first part of this chapter, we raise some well-known task scheduling methods.

On the other hand, in order to deal with the energy-efficient issue in the system level, reducing the supply voltage is considered as an efficient way since the electric power consumed per cycle with CMOS circuitry scales quadratically to the supply voltage. Therefore, we introduce some researches which concentrated on reducing power consumption by scaling supply voltages in the second part of this chapter.

Moreover, we introduce some other sensor scheduling methodologies and the battery recovery effect which affect the lifetime of batteries in the third part and the fourth part of this chapter, respectively.

2.1 Periodic Task Scheduling

A periodic task [2] is a task that has a constant time interval between successive task request times. In this section, we show two basic scheduling algorithms for handling periodic tasks: Earliest Deadline First (EDF) scheduling algorithm and Rate Monotonic (RM) scheduling algorithm. Moreover, a test that determines whether a set of ready tasks can be scheduled such that each task meet its deadline is called a schedulability test. We show the schedulability analysis of each scheduling algorithm.

In order to simplify the schedulability analysis, the following assumptions were made on the tasks [2]:

- A1: The instance of a periodic task τ_i are regularly activated at a constant rate. The interval T_i between two consecutive activations is the period of the task.
- A2: All instances of a periodic task τ_i have the same worst case execution time C_i.
- A3: All instances of a periodic task τ_i have the same relative deadline D_i, which is equal to the period T_i.
- A4: All tasks in a periodic task set Γ are independent; that is, there are no precedence relations and no resource constraints.

In addition, the following assumptions are implicitly made:

- A5: No task can suspend itself, for example on I/O operations.
- A6: All tasks are released as soon as they arrive.
- A7: All overheads in the CPU are assumed to be zero.

Before introducing the scheduling algorithms, we would like to introduce the principle of "preemption" and "non-preemption" [3]. With priority-based scheduling, a high priority task may be released during the execution of a lower-priority task. In a preemptive scheme, there will be an immediate switch to the higher-priority task. While with non-preemption, the lower-priority task will be allowed to complete before the other executions. In general, preemptive schemes enable higher-priority tasks to be more reactive, and hence they are preferred. Between the extremes of preemption and non-preemption, there are alternative strategies that allow a lower-priority task to continue to execute for a bounded time (but not necessarily to completion). These schemes are known as deferred preemption or cooperative dispatching.

2.1.1 Earliest Deadline First Scheduling

The Earliest Deadline First (EDF) [2][4] algorithm is a dynamic scheduling rule that selects tasks according to their absolute deadlines. EDF forms a priority queue where the tasks with earlier deadline will be executed at higher priorities. Although it is usual to know the relative deadlines of each task in some applications (for example 100ms after release), the absolute deadline are computed at run-time, and hence EDF is described as a dynamic scheduling.

Table 2.1: A periodic task set

Task	Computation time C	Period T
τ_1	1	5
τ_2	2	7
τ_3	4	8

In [5], Liu and Layland showed that by considering only the utilization of the task set, a test for schedulability can be obtained. Thus, the schedulability test of EDF is defined as:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

where the C_i are the worst-case computation-times of the n tasks and the T_i are their respective inter-arrival periods which were assumed to be equal to the relative deadlines.

For example, considering a periodic task set shown in Table. 2.1. The utilization is

$$U = \frac{1}{5} + \frac{2}{7} + \frac{4}{8} \approx 0.99$$

This means that 99% of the processor time is used to execute the periodic tasks, whereas the CPU is idle in the remaining 1%. Since the utilization is less than 1, the task set is schedulable by using EDF.

2.1.2 Rate Monotonic Scheduling

The Rate Monotonic (RM) scheduling algorithm [2][4] is a simple scheduling rule that assigns priorities to tasks according to their request rate. In RM scheduling algorithm, tasks with higher request rate will have higher priorities. Since the periods of tasks are constant, RM scheduling is a fixed-priority assignment that priorities are assigned to tasks before execution and do not change over time. Moreover, RM scheduling is intrinsically preemptive, the currently executing task is preemptive by a newly arrived task with shorter period. Liu and Layland [5] showed that RM scheduling is optimal among all fixed-priority assignment in the sense that no other fixed-priority scheduling algorithms can schedule a task set that cannot be scheduled by RM scheduling algorithm.

The schedulability test defined for RM scheduling algorithm is:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$$

where the C_i are the worst-case computation-times of the n tasks and the T_i are their

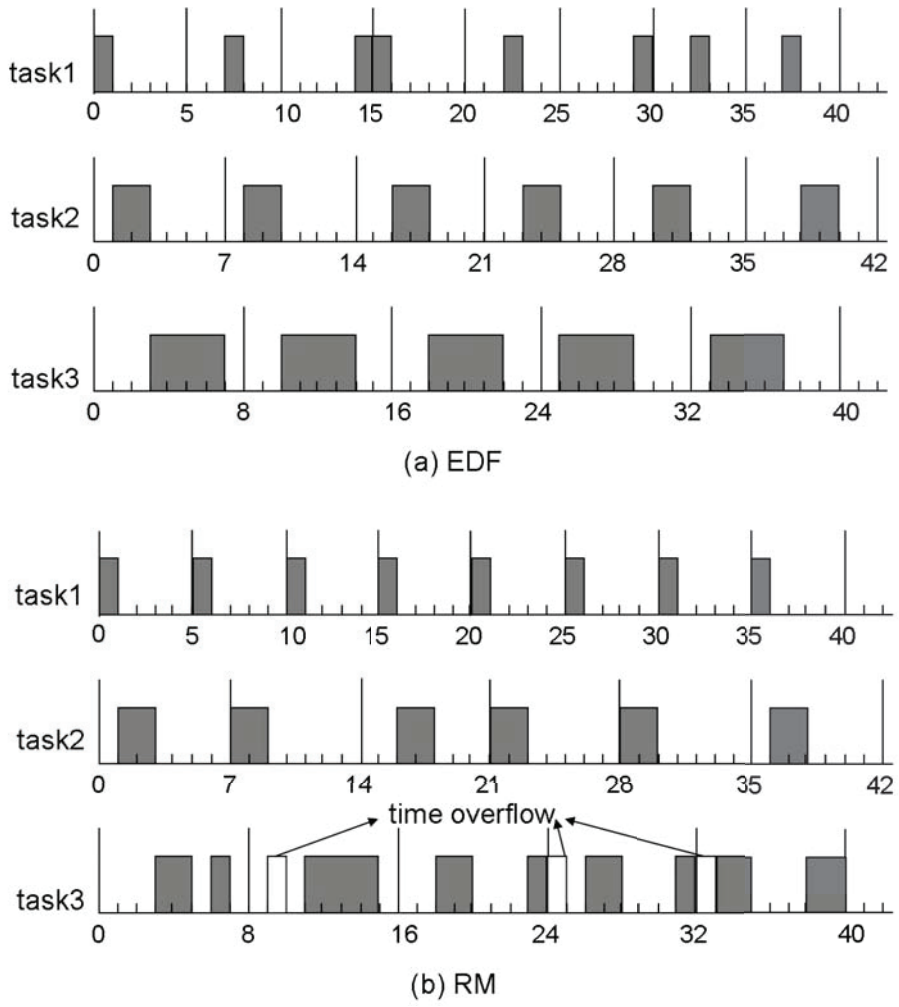


Figure 2.1: Schedule produced by EDF and RM on the same set of periodic tasks

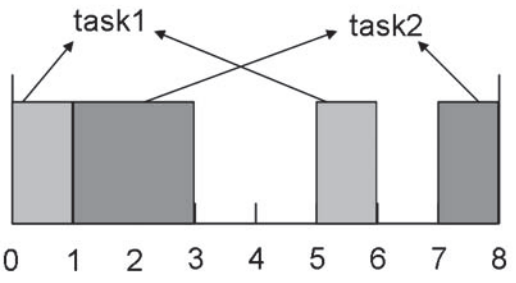


Figure 2.2: CPU utilizations before scheduling task 3

respective inter-arrival periods. We show a schedulability test of RM scheduling algorithm by using Table. 2.1. The utilization is

$$U = \frac{1}{5} + \frac{2}{7} + \frac{4}{8} \approx 0.99$$

However,

$$n(2^{\frac{1}{n}}-1)\approx 0.78$$

which is smaller than the utilization when three tasks exist in a task set. Hence, the schedulability of the task set can not be guaranteed under RM scheduling algorithm, whereas it is guaranteed under EDF. As shown in Fig. 2.1, EDF completed all tasks within their deadlines (Fig. 2.1(a)), whereas RM generates three time-overflows at time 9, 25 and 32 (Fig. 2.1(b)). The reason why RM may generates time-overflows in task 3 is that since task 1 and task 2 has higher priorities than task 3, there is not enough time slots for scheduling task 3 in one period after scheduling task 1 and task 2. As shown in Fig. 2.2, the CPU has been utilized 5 time-slots after scheduling task 1 and task 2 in a time of 8 period (one period in task 3), however, the period only remains 3 time-slots which is not enough for executing task 3 in this period. Therefore, task 3 will be executed 3 time-slots in this period, and hence generate one time-overflow in next period since task 3 occupies four time-slots. Generally, since

$$\lim_{n\to\infty} n(2^{\frac{1}{n}}-1)=\ln 2\approx 0.693$$

a rough estimate for RM scheduling algorithm in the general case that a task set can meet all the deadlines if CPU utilization is less or equals to 69.3%.

2.2 Energy-Efficient Approaches

Generally, battery power is dissipated in two ways: dynamic dissipation, in which power is consumed by CPU and devices, and static dissipation, which is leakage power. Since most of the power is consumed by the dynamic power dissipation, a lot of research has focused on how to reduce it from circuit and hardware level to system software to application level (eg. [6]). In this section, we will introduce some typical approaches in reducing power dissipations in a system level design.

2.2.1 Dynamic Voltage Scaling

In [7], Burd and Brodersen pointed out that Scaling clock frequency coupled with supply voltage is the most beneficial for energy-efficiency in microprocessor design. The authors devised an algorithm called Dynamic Voltage Scaling (DVS) that reduces the supply voltage by enabling the CPU to operate at a lower frequency. In nowadays, Dynamic voltage scaling (DVS) becomes a commonly-used technique to save power on a wide range of computing systems, from embedded, laptop and desktop systems to high-performance server-class systems.

In CMOS technology, the maximum operating frequency increases with increased operating voltage, the faster the processor, the higher the energy costs. Therefore, when the processor runs lower, a reduced operating voltage suffices, and low power consumption can be obtained. In [7], the authors defined a metric for energy consumption in the CMOS:

$$Power = V_{DD}^2 * f_{CLK} * C_{EFF}$$

where V_{DD} is the supply voltage in a circuit, f_{CLK} is clock frequency and C_{EFF} is the effective switched capacitance. The formula shows that the power dissipated per cycle with CMOS circuitry scales quadratically to the supply voltage ($E \propto V^2$). It means even a small change in voltage can have a significant impact on energy consumption. By dynamically scaling both supply voltage and frequency of the processor, DVS can provide the reduction of power consumption.

2.2.2 Real-Time Dynamic Voltage Scaling

We illustrated Dynamic Voltage Scaling (DVS) in the last subsection. However, for a large class of applications in embedded real-time systems such as digital cameras and smart phones where tasks must be accomplished by some specified deadlines, the variable operating frequency interferes with their deadline guarantee mechanisms, and DVS despites this important mechanism. Okuma et al. [8] formulated minimizing of power consumption in real-time systems as an integer liner programming and proposed scheduling algorithms in cases of known and unknown task arrival times. Moreover, Real-Time Dynamic Voltage Scaling (RT-DVS) [9] is presented as a appropriate solution that modify the OS's real-time scheduler and task management service to provide significant energy savings while maintaining real-time deadline guarantees.

In RT-DVS [9], the authors developed three algorithms to integrate DVS mechanisms into Earliest-Deadline-First (EDF) and Rate Monotonic (RM) schedulers.

Static Voltage Scaling

The authors first proposed a very simple mechanism for providing voltage scaling while maintaining real-time schedulability called "static voltage scaling." In this mechanism, the authors pick the lowest possible operating frequency that will allow the RM of EDF scheduler to meet all the deadlines for a given task set. This frequency is set statically, and will not be changed unless the task set is changed. For selecting the appropriate frequency, the aothors observed that scaling the operating frequency by a factor α ($0 < \alpha \leq 1$) may results in the worst case computation time by a task scaled by a factor $1/\alpha$, while the desired period and deadline remains unaffected. And

$$\alpha = f_i/f_m,\ where\ f_i \in \{f_1, \cdots, f_m\}, i = 1, \cdots, m$$

The "static voltage scaling" uses the schedulability test to test if a task set can be scheduled by using the scaled values for worst-case computation needs of the tasks. Moreover, the authors assumed that the operating frequencies and the corresponding voltage settings available on the particular hardware platform are specified in a table provided to the software. Therefore, if the EDF schedulability test satisfies:

$$C_1/T_1 + \cdots + C_n/T_n \leq \alpha$$

the task set can be scheduled by EDF by using a lowest frequency f_i. Moreover, if the RM schedulability test satisfies:

$$\lceil T_i/T_1 \rceil * C_1 + \cdots + \lceil T_i/T_i \rceil * C_i \leq \alpha * T_i$$

the task set can be scheduled by RM by using a lowest frequency f_i.

Cycle-Conserving RT-DVS

Although real-time tasks are specified with the worst-case computation requirements, they generally use much less than the worst case on most invocations. Thus, a DVS mechanism could reduce the operating frequency and voltage when tasks use less than their worst-case time allotment, and increase frequency to meet the worst-case needs. However, under a worst-case computation assumption, the tasks may complete earlier by comparing the actual processor time used to the worst-case specification. Thus, instead of idling for extra processor cycles, the authors devised their second algorithm "cycle-conserving RT-DVS" that avoid wasting cycles (hence "cycle-conserving") by reducing the operating frequency.

For EDF scheduling, if a task completes earlier than its worst-case computation time, "cycle-conserving RT-DVS" reclaim the excess time by recomputing utilization using the actual computing time consumed by the task. This reduced value is used until the task is released again for its next invocation. Therefore, "cycle-conserving RT-DVS" select frequency by using the EDF schedulability test:

$$U_1 + \cdots + U_m \leq \alpha$$

and set U_i to C_i/T_i upon the task releases, whereas set U_i to cc_i/T_i upon the task comletes, where cc_i is the actual cycles used in this invocation.

However, to form a “cycle-conserving RT-DVS” for RM scheduling, the schedulability test is more complex, and the complexity is n^2 where n is the number of tasks to be scheduled. The authors showed a different approach in RM scheduling as in below. Firstly, “cycle-conserving RT-DVS” initially start with worst-case schedule base on static scaling, in this step, the algorithm uses the maximum frequency in the discrete frequency set. Secondly, the algorithm determines minimum frequency so as to complete the same task by the first deadline in order to spread out the work that should be accomplished before this deadline. Thirdly, after executing the first task, the algorithm repeat the exercise of spreading out the remain work over the remaining time until the next deadline, which results in a lower operating frequency since the first task completed earlier than its worst-case specified computing time. This procedure repeating at each scheduling point until the last task is scheduled.

Look-Ahead RT-DVS

The final RT-DVS algorithm that the authors proposed attempts to achieve even better energy saving using a look-ahead technique to determine future computation need and defer task execution, hence, it is called “look-ahead RT-DVS.” The look-ahead scheme tries to defer as much work as possible, and sets the operating frequency to meet the minimum work that must be done immediately to ensure all future deadlines are met. Although this may forces tasks run at high frequencies later in order to complete all of the deferred work in time, the authors notice that if tasks tend to use much less than their worst-case computing time allocations, the peak execution rates for deferred work may never be needed. Therefore, the algorithm will allow the system to continue operating at a low frequency and voltage while completing all tasks by their deadlines.

The authors explained how a look-ahead RT-DVS EDF algorithms works. The goal is to defer work beyond the earliest deadline in the system so that tasks can operate at a low frequency now. Firstly, the algorithm allocates time in the schedule for the worst-case execution of each task, starting with the task with the latest deadline. Secondly, the algorithm spread out the work of the task with latest deadline between the first deadline and the latest deadline, subject to a constraint reserving capacity for future invocations of the other tasks. Thirdly, repeat the second step for the tasks with the latest deadline except the tasks which have been dealt. The algorithm use the work allocated before the first deadline to determine the operating frequency. Once the first task has completed, using less than its specified worst-case execution cycles, the algorithm repeat this and find a lower operating frequency. Continuing this method of trying to defer work beyond the next deadline in the system, the final results in the execution trace will be obtained.

2.2.3 Quasi-Static Voltage Scaling

In [10] and [11], the authors mentioned that although DVS and body-biasing can significantly reduce the dynamic and static energy dissipation of systems, DVS is computationally expensive, and hence significantly hampers the possible energy savings. In order to take full advantage of slack that arises from variations in the execution time, it is important to recalculate the voltage setting during run-time, i.e. on-line. Therefore, the authors proposed a Quasi-Static Voltage Scaling (QSVS) [10] [11] algorithm with a constant online time complexity of $O(1)$, which allows to increase the exploitable slack as well as to avoid the energy dissipated due to on-line recalculation of the voltage settings. QSVS selects optimal voltage setting by solving the linear problem proposed in their paper [12]. Moreover, O. Jovanovic [13] proposed an off-line and on-line technique to readjust the voltage settings at run-time for multi-processor real-time systems.

The basic idea of QSVS is divided into two phases. In the first phase, which is performed before the actual execution (i.e., off-line), voltage settings for all tasks are precomputed based on possible task start times. The resulting voltage/frequency settings are stored in look-up tables (LUTs) that are specific to each task. It is important to note that this phase performs the time intensive optimization of the voltage settings. The second phase is performed on-line. Each time new voltage settings for a task need to be calculated, the on-line scheme looks up the voltage/frequency settings from the LUT based on the actual task start time. If there is no exact entry in the LUT that corresponds to the actual start time, then the voltage settings are estimated using a linear interpolation between the two entries that surround the actual start time.

Off-line Algorithm

Consider a set of NT tasks, $T = \{\tau_i\}$ such that the execution order is fixed according to an EDF policy. Each task τ_i is characterized by a six-tuple,

$$\tau_i = < BNC_i,\ ENC_i,\ WNC_i,\ Ceff_i,\ D_i >$$

where BNC_i, ENC_i and $\mathrm{WNC_i}$ denote the best-case, the expected-case and the worst-case number of clock cycles, respectively, that task τ_i requires for its execution. Furthermore, $Ceff_i$ and D_i represent the effective switched capacitance and the deadline.

The off-line algorithm is shown in Table 2.2. Upon initialization, the algorithm computes the earliest and latest start times for each task (lines 01-06). The algorithm proceeds

Table 2.2: Off-line Algorithm [11]

```
Algorithm: $QUASI\_STATIC\_VS\_OFFLINE$
Input:  - execution order of tasks $\tau \in T$
        - for all tasks $\tau_i \in T$:
          $BNC_i, ENC_i, WNC_i, Ceff_i, D_i$
        - NL
Output: - Lookup tables $LUT_i$
01 for $i = 1$ to $NT$ {
02     $EST_i \leftarrow calc\_earliest\_starttime$
03 } // end for
04 for $i = NT$ downto 1 {
05     $LST_i \leftarrow calc\_latest\_starttime$
06 } // end for
07 $T_r \leftarrow T$
08 for all $\tau_i \in T$ { // $i = 1...NT$
09     $I_i \leftarrow LIST_i - EIST_i$
10     $j \leftarrow 0$
11     $n_i \leftarrow comp\_interpolation\_points(\tau_i, LST_i, EST_i)$
12     for $(t_s \leftarrow EST_i;\ t_s \leq LST_i;\ t_s \leftarrow t_s + I_i/n)$ {
13         $t_{s_i} \leftarrow t_s$
14         $(Vdd_i,\ Vbs_i,\ f_i) \leftarrow volt\_scaling(T_r, t_{s_i})$
15         $LUT_i[j] \leftarrow store\_QS\_lookup(t_{s_i},\ Vdd_i,\ f)$
16         $j \leftarrow j + 1$
17     }
18     $T_r \leftarrow T_r - \tau_i$
19 } // end for all
20 for all $\tau_i \in T$ return $LUT_i$
```

by initializing the set of remaining tasks T_r with the set of all tasks T (line 07). In the following (lines 08-18), the voltage and frequency settings for the start time intervals of each task are calculated. The voltage scaling algorithm used in line 14 can be formulated as a convex nonlinear optimization as follows:

Minimize

$$\sum_{k=i}^{|T_i|} E_{dyn_k} + E_{leak_k}$$

where

$$E_{dyn_k} = ENC_k * C_{eff_k} * V_{dd_k}^2$$

and

$$E_{leak_k} = L_g(K_3 * V_{dd_k} * e^{K_4 * V_{dd_k}} * e^{K_5 * V_{bs_k}} + I_{Ju} * \ V_{bs_k}\ ) * t_k$$

subject to

$$\begin{cases} t_k = ENC_k * \dfrac{(K_6 * L_d * V_{dd_k})}{((1+K_1) * V_{dd_k} + K_2 * V_{bs_k} - V_{th_1})^{\alpha}} \\ sk + t_k \leq s_{k+1} \\ sk + t_k \leq D_k \ \forall \tau_k \text{ that have a deadline} \\ sk + t_k \leq LST_{k+1} \\ s_k > 0 \\ V_{dd_{min}} \leq V_{dd_k} \leq V_{dd_{max}} \text{and } V_{bs_{min}} \leq V_{dd_k} \leq V_{bs_{max}} \end{cases}$$

The variables that need to be optimized are the task execution times t_k, the task start time s_k as well as the supply voltage V_{dd_k} and body bias voltage V_{bs_k}. With our best knowledge, this formulation can be solved by polynomial-time. This on-line algorithm returns the quasi-static scaling table LUT_i for all tasks.

On-line Algorithm

Table 2.3: On-line Algorithm [11]

```
Algorithm: $QUASI\_STATIC\_VS\_OFFLINE$
Input:   - start time $t_{s_n}$ of next task $\tau_n$
         - Quasi-Static Scaling Table $LUT_n$
           number of start time interval steps n
         - frequency and voltage settings for task $\tau_n$
Output: - Lookup tables $LUT_i$
01  $(x,\ y) \leftarrow calc\_st\_interval(LUT_n, t_{s_n})$
02  $f_n \leftarrow inter\_freq(LUT_n, x, y, t_{s_n})$
03  $Vdd_n \leftarrow inter\_Vdd(LUT_n x, y, t_{s_n})$
04  $Vbs_n \leftarrow calc\_Vbs(f_n, Vdd_n)$
05  **return** $(f_n, Vdd_n, Vbs_n)$
```

The on-line algorithm is shown in Table 2.3. This algorithm is called each time after a task finishes its execution, in order to calculate the voltage settings for the next task τ_n. In the first step, the algorithm calculates the two entries x and y from the quasi-static scaling table LUT_n that contain the start times which surround the actual time t_{s_n} (line 01). According to the identified entries, the frequency setting f_n for the execution of task τ_n is linearly interpolated using the two frequency settings from the quasi-static scaling table $LUT_n[x]$ and $LUT_n[y]$ (line 02). Similarly, in line 03 the supply voltage Vdd_n is linearly interpolated from the two surround voltage entries in LUT_n. Line 04 calculates the body-bias voltage directly for the interpolated frequency and supply voltage values, using the following equation:

$$f = \frac{((1+K_1)*V_{dd_k}+K_2*V_{bs_k}-V_{th_1})^{\alpha}}{(K_6*L_d*V_{dd_k})}$$

Finally, the algorithm terminates and returns the settings for the frequency, supply and body-bias voltage (line 05).

2.3 Sensor Scheduling

In Section 2.1 and 2.2, we introduced some task scheduling algorithms and three voltage/frequency scaling algorithms in order to reduce energy dissipation in embedded systems. However, all those algorithms were concentrated on how to minimizing the power consumption of processors, there has been little research on reducing the power consumption of devises, such as sensor nodes.

In [14], A. Krause et al. mentioned that when deploying sensor networks for monitoring tasks, both placing and scheduling the sensors are important in order to ensure informative measurement and long deployment life. Therefore, they proposed a near-optimal approach that simultaneously optimized the placement and scheduling of sensor nodes in a wireless sensor network. In their approach of scheduling sensors, they separated sensors into several “buckets,” with each “bucket” having a certain number of sensors that compose a big element or a small element. They defined a function to judge whether the bucket conditions are satisfied by the sensing quality or not. If the bucket conditions are not satisfied, the sensors composing the big element will be reallocated to other buckets until all bucket conditions are satisfied. A. Krause et al. did not consider a deadline, and they assumed the sensors are simultaneously activated. Due to our best knowledge, simultaneous sensor activation is significantly power consuming. Thus, we will make a effort to solve the sensor scheduling problem in the following sections.

2.4 Battery Recovery Effect

In the above sections, we introduced some energy efficient approaches in embedded systems which use batteries to provide energies. In battery-powered systems, battery behaviors also can influence the system life since the rate capacity effect and recovery effect of batteries exist. Therefore, we will introduce some basic knowledges about battery and some battery model in this section.

The battery lifetime mainly depends on the rate of power consumption of the device. In the ideal case, the battery supply voltage stays constant during discharge and the battery

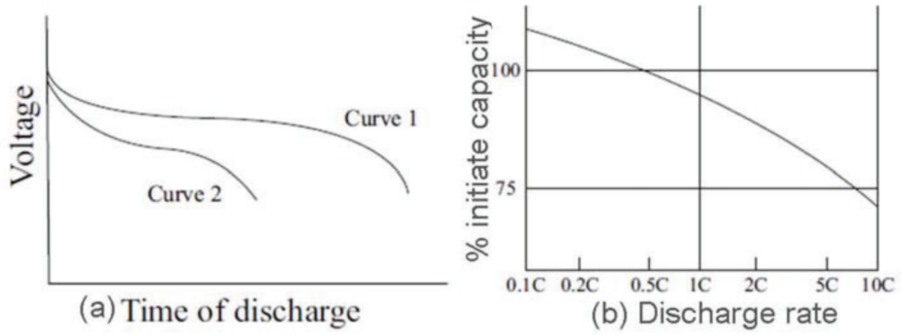


Figure 2.3: Rate capacity effect [15]

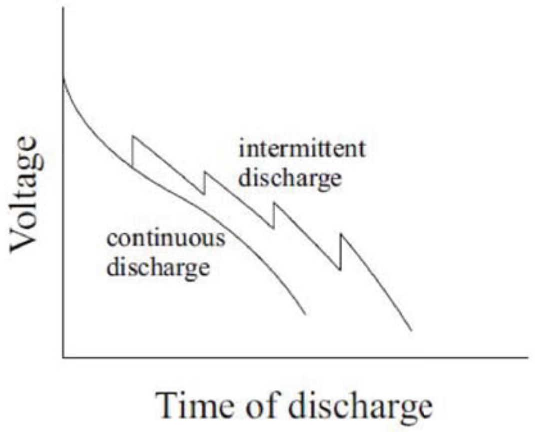


Figure 2.4: Recovery effect [15][16]

capacity would be constant for all discharge current; all energy stored in the battery would be used with the battery voltage drops to zero. However, for a real battery, the behaviors of the capacity and voltage drop are non-linear because some chemical and physical factors. These behaviors were termed rate capacity effect and recovery effect [15][16]. We use Fig. 2.3 to illustrate the rate capacity effect. Fig. 2.3(a) shows the evolution of the voltage over time for a low and high discharge current, curve 1 and 2, respectively. The voltage drops faster for high discharge currents. Fig. 2.3(b) shows the capacity as a function of the discharge rate, and effective capacity drops for high discharge rate. The discharge rate is given in terms of "C rate," a C rate of nC means that the battery is discharged in $1/n$ hours. Moreover, Fig. 2.4 shows the recovery effect. When a battery stands idle after a discharge (intermittent discharge), some chemical and physical reactions happen which result in a recovery of the battery voltage. Thus the voltage of a battery, which has dropped during a heavy discharge, will rise after a rest period, giving a sawtooth shaped discharge, and this will result in an increase of battery life. Since in addition to current drain, the extent of recovery is dependent on many other factors such as the particular battery system and constructional features, discharge temperature, operation voltage, and

Table 2.4: Battery models overview [15]

model	rate capacity effect	recovery effect	accuracy
Dualfoil [17]	+	+	very high
Peukert [18]	+	-	medium, 10% error
Rakhmatov [18]	+	+	high, 5% error
Chiasserini [19]	-	+	high, 1% error

length of recovery period, the recovered voltage is very complicate to be calculated.

Moreover, in [15], the authors summarized some well-known battery models. Since the battery used in the sensor nodes in this thesis is a Li-ion type battery, thus, we only show some battery model which can deal Li-ion type batteries in Table. 2.4. By using these battery models, it will be possible to model the power consumption of battery-powered devices, and predict battery lifetimes for different usage patterns.

Chapter 3

Problem Definition Of Sensor Scheduling

3.1 Experimental Results with General Sensor Execution Schedule

As we indicated the problem in chapter 1, the problem is that the battery charge still remains when the battery voltage reaches the operation voltage. Before beginning this study, we had developed a prototype energy management platform called "Green Tap" [20][21] (see Fig. 3.1), which consists of a wireless station and wireless sensor nodes comprised of a CPU, a button battery, ZigBee device, and various environment sensors such as a motion sensor, an illumination sensor, a temperature sensor, and a humidity sensor. To investigate its aspects, we measured the voltage drop, battery voltage, and battery charge of sensors in a node in experiments where sensors operated according to two schedules. The sensor node we used is CP04 (only used in experiments, not for sale) made by NEC, and the button battery we used is CR2450 made by Panasonic. The specifications of the battery and the sensor node are shown in Table 3.1 and Table 3.2, respectively.

Schedule A: Sensors run for as short a time as possible, and the CPU is kept asleep for as long as possible (Fig. 3.2(a)). In this case, the sensors are activated at the same time.

Schedule B: Sensors run with as much delay as possible, and the CPU sleeps for as short a time as possible (Fig. 3.2(b)). In this case, the sensors are sequentially activated.

The results are shown in Fig. 3.3. When the sensor ran according to schedule A, the battery voltage reached the operation voltage (2.5 V) after 5000 hours, but this was before

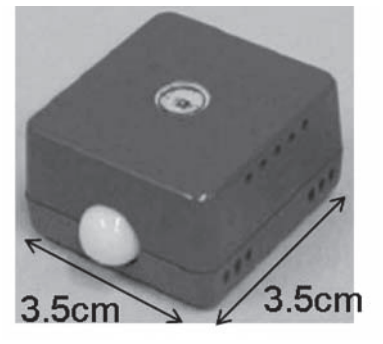


Figure 3.1: GreenTap

Table 3.1: Battery specification

Nominal voltage (V)	3
Nominal capacity (mAh)	620
Continuous standard load (mA)	0.2
Operating temperature (°C)	-30∼+60

the battery charge fell to zero. When the sensor ran according to schedule B, the battery voltage did not reach the operation voltage when the battery charge reached zero (7500 hours). This means that since the peak power consumption of sensors is large during simultaneous sensor data acquisition, battery voltage falls faster than battery charge and hence reaches the operation voltage before battery is fully discharged. Generally, this tendency is described in [16]. In [16], the authors show that the voltage drops faster for high discharge current, i.e., high peak power consumption. Moreover, in [22], the authors mentioned that peak power consumption should be reduced wherever possible, which means operations that consume a battery should be performed serially rather than concurrently. Serial operation is better than concurrent operation when each consumes roughly the same energy.

To solve this problem, we can lower the peak power consumption by controlling sensor activations. One way to control sensor activation is to sequentially activate it. How-

Table 3.2: Sensor node specification

Sensor	Measurement frequency	Measurement range	Resolution	Accuracy
motion	second	distance: 2.5m view angle: 100°	None	None
temperature	minute	-10∼50°C	1°C	±1°C
humidity	minute	20∼80%RH	1%RH	±3%RH

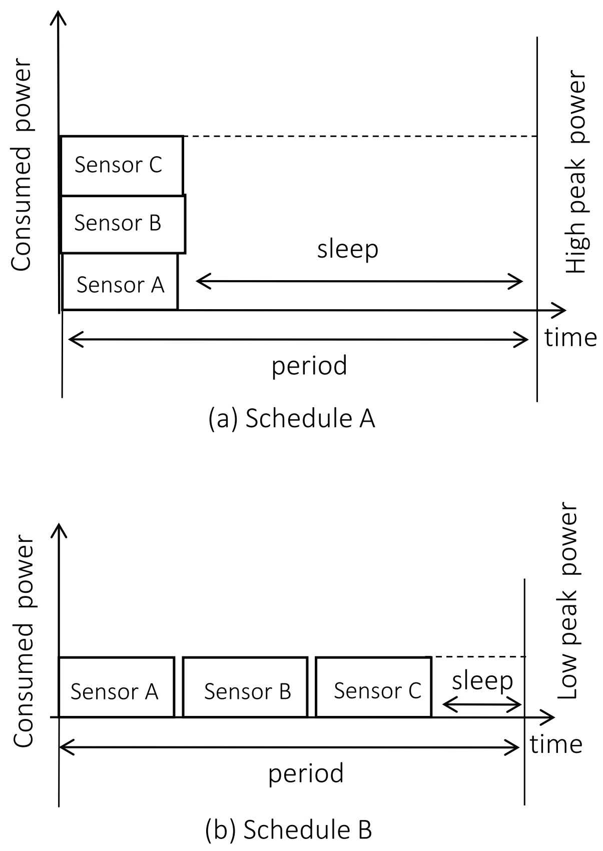


Figure 3.2: Two sensor execution schedules

ever, sensors cannot always be sequentially activated in circumstances when they have finish data acquisition before a deadline. This means we need a solution such as, ideally, the battery will run out of charge when it reaches the operation voltage with real-time constraints. As a first step toward this goal, we devised a way to decrease the peak power consumption.

3.2 Sensor Scheduling Techniques and Strategies

Supposing N sensors exist in a node, we assume that a sensor has a deadline because the sensor data is obtained within a certain time and consumes power during the activation time. One of such application that requires sensor data deadline is a power management system in consumer electronics at home using GreenTap [23]. In this system, the authors observed that sampling voltages and frequencies of electronics should be required at 50MHz and 60MHz, respectively. Another application is a health monitoring systems gathering human pulse. Therefore, sensors that belongs to a node is characterized by a relative deadline L_i, data acquisition time k_i, and power consumption constant d_i (Fig. 3.4). A sensor σ_i is periodically activated at time $T_a^{\sigma_i}$ and deactivated at time $T_d^{\sigma_i}$ in one

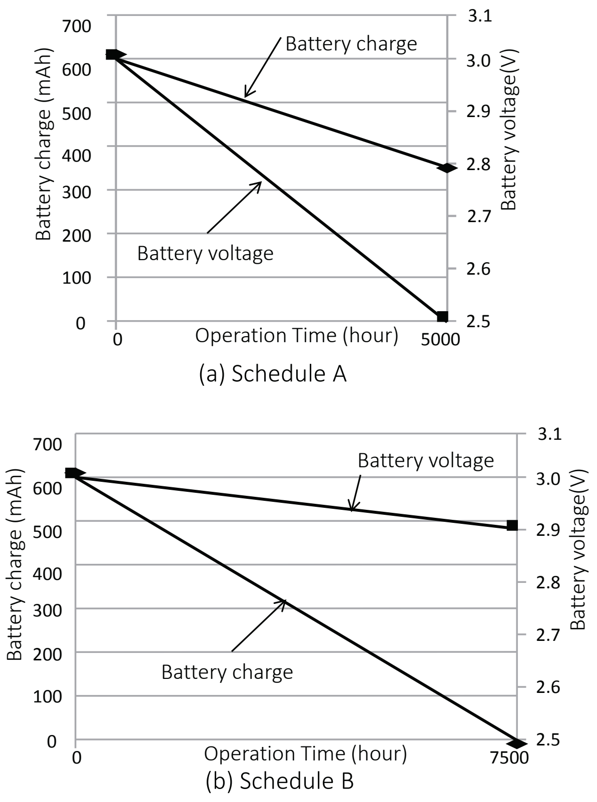


Figure 3.3: Experiment results of the two sensor schedules

time period. All of these time value are relative to the start of the time period as mention later.

The following is an example to show how to solve the problem using Fig. 3.5. We assume that three sensors are activated simultaneously, as shown in Fig. 3.5(a). Although this schedule can meet the deadline, its peak power consumption of $d_x + d_y + d_z$ causes a problem wherein the battery voltage reaches the operation voltage before the battery charge falls to zero. However, if the sensors are activated sequentially, some sensors will miss the deadline. Therefore, we devised a method that sometimes activates sensors simultaneously and sometimes activates them sequentially in order to meet each sensor's deadline (see Fig. 3.5(b)). This method can efficiently reduce the peak power consumption to $d_y + d_z$. We show how to generate this kind of schedule to reduce the peak power consumption in Chapter 4.

Generally, in each sensing data period T_P, after the sensor schedule execution time T_{SE} that sensors are executed to obtain the sensing data, the sensing data should be collectively sent to a base station periodically. In typical applications using sensing data, it is sufficient that data from sensors can be obtained within some period (e.g. [24]). We assume the duration required for data transmission is T_{DT}. There are many ways to transmit the sensing data, and we will talk about that in the second part of this thesis. Moreover, a

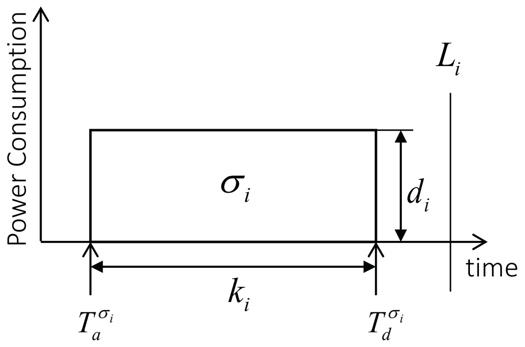


Figure 3.4: Sensor Model

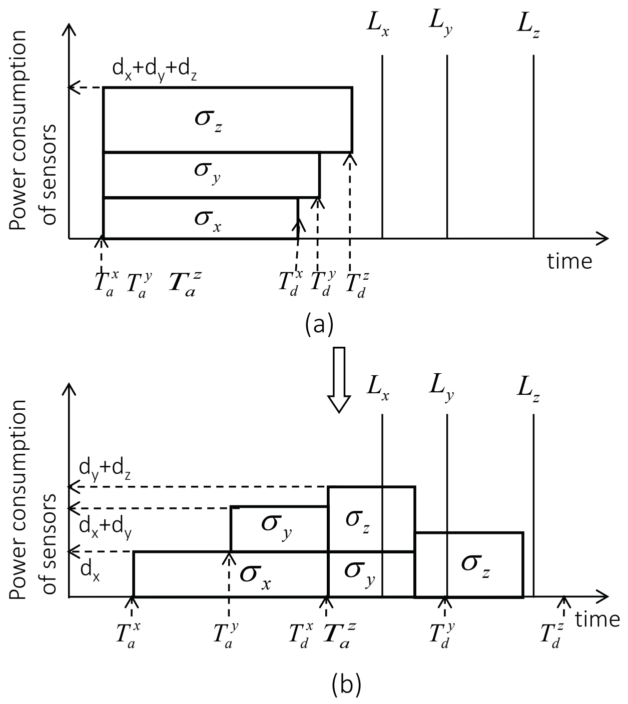


Figure 3.5: Motivation example of sensor schedule

certain sleep time T_{sleep} after all sensors were activated, is required to replenish the battery, this can result in a recovery of the battery voltage caused by intermittent discharges which is called recovery effect. This can result in an increase in battery life. However, the recovery model is quite complicated because of it depends on battery types [16, p. 3.13]. We simplify the time usage model without taking consideration of T_{DT} and T_{sleep}, and our algorithms and simulations focus on power consumption by sensor activation.

Chapter 4

Sensor and Task Scheduling Algorithms

We devised three off-line algorithms for the purpose of reducing the peak power consumption as much as possible. The first one uses the idea of EDF (Earliest Deadline First), and hence, we call it the EDF Based Scheduling Algorithm (EDF-BSA). The second schedule sensors dispersedly and try to schedule sensors below an average peak power consumption, it is named the Uniform Distribution Scheduling Algorithm (UDSA). The last one is also based on EDF, and for the reason that its feature is "fill the blanks", we called it the Blank Filling Scheduling Algorithm (BFSA). This algorithm turns out to be the most efficient for solving the problem. All three algorithms can be calculated off-line.

In chapter 3, we described the sensor scheduling problem for maximum utilization of a battery used in a sensor node. To solve it, we make the following assumptions.

- A1: The sensor's deadline L_i, data acquisition time k_i, and power consumption constant d_i are known a priori.

- A2: Each sensor must be able to operate continuously; once activated, it does not take breaks until it has finished its task.

Moreover, we make the following assumptions for simplicity.

- A3: Power consumption which consumed by CPU can be ignored.

- A4: Each sensor is independent; there is no precedence relations and no resource constrains.

Let t_k be a discrete time series where $t_{k+1} - t_k = \Delta t (k = 0, 1, ...)$, and Δt is a unit time, say 10 ms. Consider the time within a period. A period starts at time t_0, and the deadline and the other times are relative to the start time. $S(t)$ is a set of sensors activated at time t, a sensor σ_i is characterized by $< L_i, d_i, k_i > (i = 0, 1 \ldots N-1)$. We assume that a sensor σ_i is activated and deactivated at discrete times of $T_a^{\sigma_i} = t_m$ and $T_d^{\sigma_i} = t_n$, respectively. The three algorithms require the same input parameters, i.e., the sensors' character set $< L_i, d_i, k_i >$ and the number of sensors N, and they return the same output, i.e., the sensors' activation time $T_a^{\sigma_i}$ and deactivation time $T_d^{\sigma_i}$. Moreover, each period, a certain sleep time when all of the sensors are inactive, is required to replenish the battery. For simplicity, this sleep time will be ignored in the following algorithms.

4.1 EDF Based Scheduling Algorithm

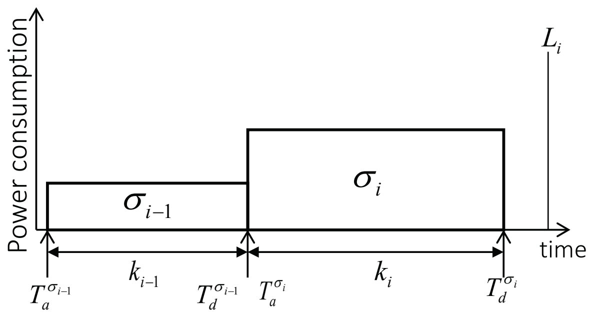
In this algorithm, we intend to schedule sensor activation in a way such that each sensor can meet its deadline based on EDF.

```
Algorithm 1: EDF Based Scheduling Algorithm
01  sort sensors by $L_i$ in increasing order
02  $T_a^{\sigma_0} = t_0$ ; $T_d^{\sigma_0} = t_0 + k_0$
03  $T_d^{\sigma_{N-1}} = L_{N-1}$ ; $T_a^{\sigma_{N-1}} = T_d^{\sigma_{N-1}} - k_{N-1}$
04  for $i$ = 1 to $N$-2
05    if $L_i - T_d^{\sigma_{i-1}} \geq k_i$
06      $T_a^{\sigma_i} = T_d^{\sigma_{i-1}}$ ; $T_d^{\sigma_i} = T_a^{\sigma_i} + k_i$
07    else
08      $T_d^{\sigma_i} = L_i$ ; $T_a^{\sigma_i} = T_d^{\sigma_i} - k_i$
```

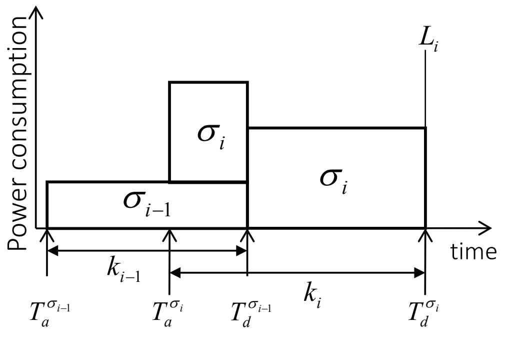
On line 01, we sort sensors by L_i in order to use EDF. We activate the first sensor at t_0, and the last sensor deactivates at its deadline on lines 02 and 03. From line 04 to line 08, we activate the other sensors at or before the deadline of the previous sensor and deactivate the sensors at or before their deadline. Figure 4.1 illustrates the procedure. In Fig. 4.1(a), after setting sensor σ_{i-1}, we set σ_i between $T_d^{\sigma_{i-1}}$ and L_i if $L_i - T_d^{\sigma_{i-1}} \geq k_i$; that is, there is enough time to accommodate activation of σ_i. Since $L_i - T_d^{\sigma_{i-1}}$ is longer than σ_i's data acquisition time k_i, we can activate σ_i at the deactivation time of σ_{i-1} to save more of the time slot after sensor σ_i's deactivates for accommodating other sensors. However, in Fig. 4.1(b), the time $L_i - T_d^{\sigma_{i-1}}$ is not long enough to activate σ_i at σ_{i-1}'s deactivation time since σ_i's deadline is earlier than in case (a). Therefore, we must activate σ_i at $L_i - k_i$ to

Table 4.1: Parameters of sensor schedule example

	σ_0	σ_1	σ_2
L	5	6	8
k	4	3	3
d	1	2	4



(a) case of $L_i - T_d^{\sigma_{i-1}} \geq k_i$



(b) case of $L_i - T_d^{\sigma_{i-1}} < k_i$

Figure 4.1: Sensor activation of EDF-BSA

meet its deadline. The complexity of EDF-BSA is $O(N)$ since the procedure is done once for each sensor.

The following example illustrates how this algorithm works. We assume there are three sensors to be scheduled (Table 4.1). Figure 4.2 shows that using the EDF-BSA, we activate σ_0 at t_0 and deactivate σ_2 at its deadline. Since $L_1 - T_d^{\sigma_0} \leq k_1$, σ_1 should be deactivated at its deadline; therefore, it is activated at $L_1 - k_1$, that is t_3.

4.2 Uniform Distribution Scheduling Algorithm

We devise another algorithm UDSA (Uniform Distribution Scheduling Algorithm) which attempt to schedule sensors below an average peak power consumption.

In UDSA, we calculate the average peak power consumption under the assumption that sensor activations are uniformly distributed during one period, and UDSA schedules sensor activation in a way sum of power consumption try to be below the average peak power consumption. First, we sort sensors in an increasing order of L_i. On lines 02 and 03, we calculate N_{over_ave} as the allowed average number of sensors which are overlapped if sensors are assumed to be executed in the uniformly distributed execution. d_{over_ave} is the average allowed value of d where sensors are overlapped. We introduce d_{over_ave}

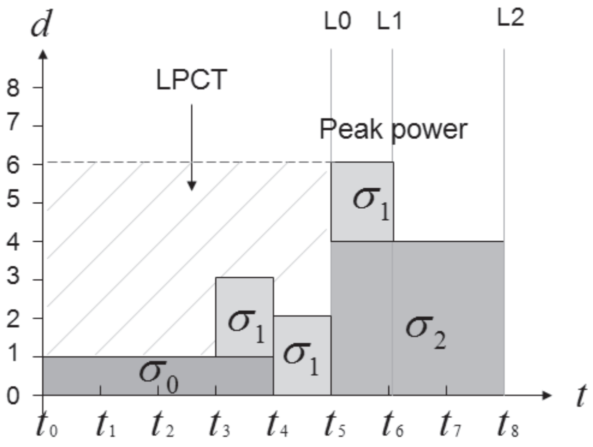


Figure 4.2: Sensor activation by EDF-BSA

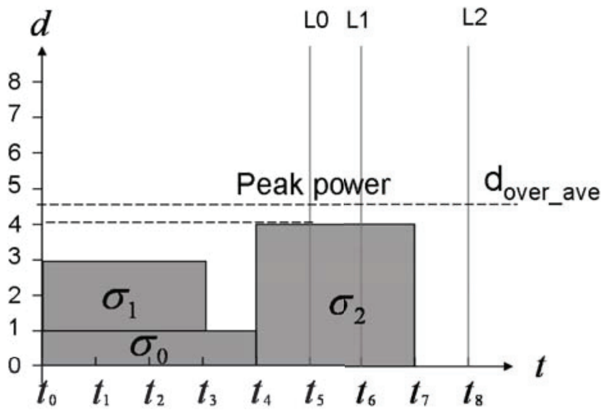


Figure 4.3: Sensor activation by UDSA

for the purpose of keeping sensors under this value as much as possible. Line 04 sets the first sensor's activation time at t_0. For the remaining sensors, UDSA activates them immediately preceding the sensor's deactivation time (line 06). Line 08 and line 09 show that if σ_i's execution cannot meet its deadline, that is $T_d^{\sigma_i} > L_i$. This algorithm attempt to activates σ_i at time when power consumption sum of σ_i and σ_l is not over d_{over_ave}, where σ_l are all of sensors simultaneously activated at the time $T_a^{\sigma_m}$, $m = i - 1$, $m \geq 0$, in order to meet σ_i's deadline (line 10). If line 10 cannot be satisfied, m decrease 1 in line 12 for moving forward σ_i's activation time to meet its deadline until the deadline can be met. Although sometimes line 10 cannot be satisfied, i.e. it may cause d to exceed d_{over_ave}, UDSA ensures the sensor meets its deadline. Thus, this " border crossing" can be allowed. Since UDSA loops N times twice, the complexity of UDSA is $O(N^2)$.

We shall use Table 4.1 and Fig. 4.3 to illustrate UDSA. d_{over_ave} is calculated as a boundary line that the overlap value of d in a time period should not go beyond. UDSA activates sensor σ_0 at t_0. If σ_1 is activated at the immediately preceding sensor's deadline,

```
Algorithm 2: Uniform Distribution Scheduling Algorithm
Variables: - $N_{over\_ave}$: the average number of sensors allowed
                 to be active simultaneously
           - $d_{over\_ave}$: the average allowed overlap of $d$
01  sort sensors by $L_i$ in increasing order
02  $N_{over\_ave} = \lceil \sum k_i / L_{N-1} \rceil$
03  $d_{over\_ave} = \sum d_i / (N / N_{over\_ave})$
04  $T_a^{\sigma_0} = t_0$ ; $T_d^{\sigma_0} = t_0 + k_0$
05  for $i$ = 1 to $N - 1$
06      $T_a^{\sigma_i} = T_d^{\sigma_u}$ ; $T_d^{\sigma_i} = T_a^{\sigma_i} + k_i$ ;
        where $T_d^{\sigma_u}$ is the latest sensor deactivation time in $\sigma_0, \ldots \sigma_{i-1}$
07      $m = i - 1$ ;
08      while $T_d^{\sigma_i} > L_i$
09          $T_a^{\sigma_i} = T_a^{\sigma_m}$ ; $T_d^{\sigma_i} = T_a^{\sigma_i} + k_i$ ;
10          if $\sum_{\sigma_l \in S(T_a^{\sigma_m})} d_l + d_i \le d_{over\_ave}$ and $T_d^{\sigma_i} \le L_i$; $where\ m \ge 0$
11              break;
12          else $m - -$
```

it will miss its deadline. From line 07 to line 11, since the $d_0 + d_1 \le d_{over\_ave}$, σ_1 should be activated at the activation time of σ_0. After locating σ_1, line 06 indicates that σ_2 should be activated at the latest deactivation time of the prior sensors, that is, $T_d^{\sigma_0}$. Moreover, σ_2 will not miss its deadline. This is the well-behaved case. We will show some simulation results of more general cases later.

4.3 Blank Filling Scheduling Algorithm

BFSA is divided into two steps. The first step is to schedule sensors based on the proposed algorithm EDF-BSA. This step yields 'blank' durations, where some sensors are allowed to be simultaneously executed. The second step is to re-schedule the sensors' active periods by shifting their activation times to the blank durations in order to reduce peak power consumption. The blank duration is called LPCT (Lower Power Consumption Time). We shift the activation time of a sensor to an LPCT to reduce peak power consumption.

The first step on line 01 calls the EDF-BSA function. The second step is from line 02 to line 15. Lines 03 and 04 indicate that for each time u_q, where u_q is an activation or deactivation time of a sensor in the revised time set $U = \{u_0, u_1, \ldots\}$, BFSA calls a method $d_{u_q} = D(u_q)$. $D(t) = \sum_{\sigma_i \in S(t)} d_i$ calculates the overlap value of d in every $T_a^{\sigma_i}$ and $T_d^{\sigma_i}$; therefore, d_{u_q} is the overlap value of d at a sensor's activation or deactivation time u_q. Lines 05 and 06 select the maximum value of d_{u_q} and pass the value of the maximum d_{u_q} to d_{max}. To reduce peak power consumption as much as possible, line 07 finds the sensor

```
Algorithm 3: Blank Filling Scheduling Algorithm
Variables: - $flag$ : a flag for recording whether a sensor's
                 activation time can be shifted or not
           - $U$ :a set of all sensor activations and deactivation time
                 in increasing order. $U = \{u_0, u_1, ...\}$
           - $u_q$ : the activation or deactivation time of a sensor,
                 where each $u_q \in U$
           - $d_{max}$ : maximum overlap of $d$; the initial value is zero.
           - $C$ : the set of sensor shift candidates which is composed of
                 the peak power consumption
           - $t'$ : a time point belonging to $t_k$
           - $t''$ : an optional time point
Functions: - EDF-BSA(): first step schedule provided by EDF-BSA
           - $D(t) = \sum_{\sigma_i \in S(t)} d_i$: sum of sensors' $d$ values at time $t$
01 EDF-BSA($T_a^{\sigma_i}$ , $T_d^{\sigma_i}$) ; flag = true
02 while flag is true
03     for each $u_q \in U$
04         $d_{u_q} = D(u_q)$
05         if $d_{u_q} > d_{max}$
06             $d_{max} = d_{u_q}$
07     $C = \{\sigma_m : \sigma_m \in S(u_q), \sigma_m \neq \sigma_0, \sigma_{N-1}\}$ in
       decreasing order of $d_m$
08     for $\sigma_l \in C$
09         find $t'$ such that $\forall t'' \in [t', t' + k_l]$: $D(t'') + d_l < d_{max}$
10             if $t'$ exist
11                 $T_a^{\sigma_l} = t'$ ; $T_d^{\sigma_l} = T_a^{\sigma_l} + k_l$ ; break;
12             else
13                 do nothing
14     if all $D_(t'') + d_l > d_{max}$
15         flag = false
```

shift candidates C that consumed d_{max}. However, C does not include the first and the last sensor since shifting these two sensors would not result in an reduction in peak power consumption. Furthermore, the sensor shift candidates are sorted in decreasing order of d for the purpose of reducing the peak power consumption as much as possible at the earlier shit procedure; that is, it shifts the sensor which has the largest value of d belonging to C first. Line 08 and 09 find out if LPCT exists by checking whether a shift in sensors can be accommodated, provided that the total power consumption is not beyond d_{max}. Note that due to a sensor's continuity, one sensor cannot be shifted separately. Lines 10 to line 13 mean that if LPCTs exist, a sensor is shifted, but if there are none, a sensor will not be shifted. The re-scheduling procedure on lines 14 and 15 stops if no more sensor shifting candidates can be accommodated by the LPCT, provided that the total power consumption

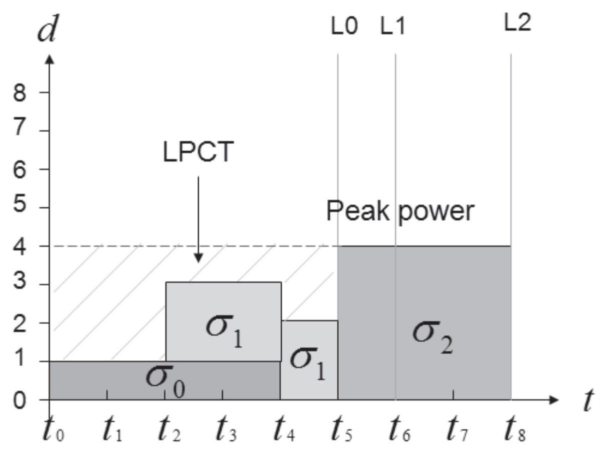


Figure 4.4: Sensor activation by BFSA

is under d_{max}. Since BFSA loops from 0 to N approximately three times, the complexity of BFSA is $O(N^3)$.

Figure 4.4 shows an example of BFSA using the parameters of Table 4.1. We use the first step schedule of EDF-BSA as the input of BFSA. In the second step from line 03 to line 07, we calculate $d_{max} = 6$ continues from time t_5 to t_6, and the candidate sensor that needs to be shifted is sensor σ_1. On line 09, we find that σ_1 can be shifted to a position between t_0 and t_5 as indicated by LPCT in Fig.4.2. Without loss of the sensor's continuity and aiming to decrease the second peak power consumption as much as possible, t', as an optimal activation time of sensor σ_1, is fixed at t_2. Therefore, we can shift σ_1's activation time at t_2. After this shift, we cannot find new blanks to reduce the new peak power consumption; hence, the scheduling procedure finishes.

4.4 Task Execution Scheduling Algorithm

As we mentioned before, a task is required to activate or deactivate sensors. In traditional ways of task execution, a task τ is executed continuously to activates and deactivates a sensor σ as shown in Fig. 4.5(a). In order to reduce the power consumed by the tasks, we assume that a task τ executed within a certain time t_c is periodically executed, thereby enabling us to activate or deactivate sensors as shown in Fig. 4.5(b). If the role of the tasks is only to activate or deactivate sensors, the tasks do not need to be executed continuously during the sensor execution since the task would cost more time and power if it is executed continuously. Therefore, we assume that a task τ to activate or deactivate one sensor periodically is executed within a certain time t_c (Fig. 4.5(b)) and that a timer controls the task execution frequency. However, the task should be able to handle a situation in which several sensors are simultaneously activated or deactivated at the same time. We consider

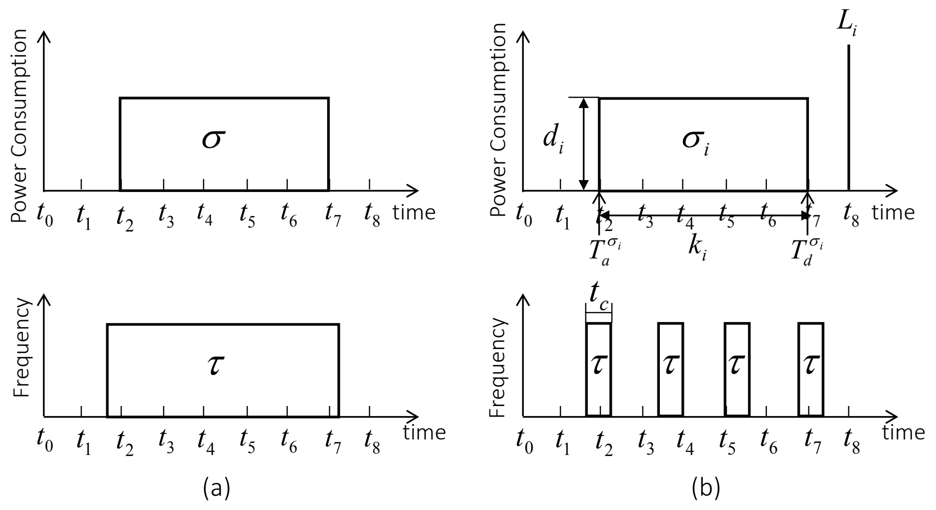


Figure 4.5: Sensor and task execution model

two methods to solve this problem. The first method is to scale the clock frequency to several times larger when the activation and deactivation times of several different sensors are the same at a time point. Moreover, the supply voltage should be scaled together with the clock frequency due to a specific DVFS ratio. In the second method, instead of clock frequency scaling, the task execution time scales to several times larger. For example, if a sensor is activated at the activation or deactivation time of another sensor at time t, then the clock frequency and the task execution time should be scaled two times larger in the first and second method at time t, respectively. Both methods are considered to be more energy efficient than continuous task execution.

Since tasks should have the capability to activate or deactivate sensors simultaneously when they are to be executed periodically, we devised two task execution scheduling algorithms to solve the problem. The first algorithm is named the DVFS-enabled periodical task execution algorithm (DVFS-PTEA); it selects the clock frequency and supply voltage and finds the scaling time point to schedule the task executions. In the DVFS concept, the supply voltages are proportioned to the corresponding clock frequencies. The second algorithm is called the execution time scaling periodical task execution algorithm (ETS-PTEA); it scales the task execution time when task scheduling problems occur. Both DVFS-PTEA and ETS-PTEA require a sensor activation and deactivation time set $U = \{u_0, u_1, ...\}$ as input. In DVFS-PTEA, the output is the scaling time point and the supply voltage and clock frequency set, which can be looked up by using a look up table based on the specific DVFS ratio. In ETS-PTEA, the output is the scaling time point and the task execution time at the scaling time point.

In DVFS-PTEA, a task is periodically executed within a certain time t_c to activate or deactivate a sensor. We assume the clock frequency f is required to execute the task. In

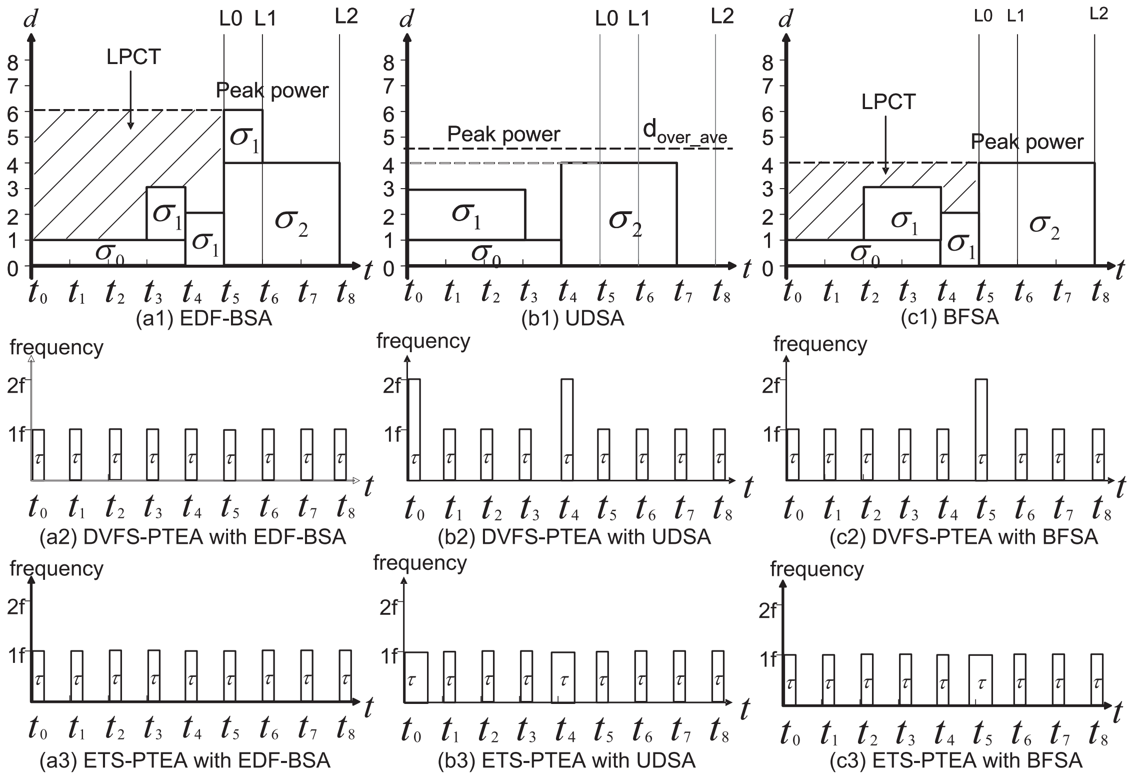


Figure 4.6: Sensor schedule and task schedule of proposed method

the task execution scheduling, DVFS-PTEA finds the time point at which w sensors were activated or deactivated simultaneously; then it scales the clock frequency to $w * f$ and scales the supply voltage due to a specific DVFS ratio. During the rest of the execution time, the task keeps on executing at the clock frequency f. In ETS-PTEA, instead of scaling the clock frequency and supply voltage, the execution time of the task scales to $w * t_c$, where w sensors were activated or deactivated simultaneously.

Let us show an example to illustrate the task execution scheduling together with the sensor scheduling. The example uses the sensor parameter set shown in Table 4.1 in section 4.1. Figure 4.6 shows the sensor schedules and the task schedules of our methods. Figures 4.6(a1), (b1), and (c1) show the sensor schedule produced by the three sensor scheduling algorithms. Figures 4.6(a2), (b2), and (c2) show the task execution schedule of DVFS-PTEA. In Fig. 4.6(a2), clock frequency does not need to be scaled to a higher frequency since there are no simultaneous sensor activations or deactivations in EDF-BSA. However, In UDSA (Fig. 4.6(b2)), the clock frequency scales to $2f$ at t_0 and t_4 since σ_0 and σ_1 are activated simultaneously and σ_0 deactivates at the activation time of σ_2. Likewise, in BFSA (Fig. 4.6(c2)), the clock frequency scales to $2f$ at t_5. Compared with scaling the clock frequency in DVFS-PTEA, ETS-PTEA scales the task execution time at the same scaling time point of that in DVFS-PTEA in Figs. 4.6(a3), (b3), and (c3).

Table 4.2: Evaluation of task scheduling algorithms in the case of three sensor scheduling algorithms

	EDF-BSA(d)	UDSA(e)	BFSA(f)	d:e:f
Continuous(a)	$22\alpha Cf^3$	$29\alpha Cf^3$	$22\alpha Cf^3$	1:1.32:1
DVFS-PTEA(b)	$0.8\alpha Cf^3$	$0.94\alpha Cf^3$	$0.87\alpha Cf^3$	1:1.75:1.09
ETS-PTEA(c)	$0.8\alpha Cf^3$	$0.82\alpha Cf^3$	$0.81\alpha Cf^3$	1:1.03:1.01
a:b:c	27.5:1:1	35.4:1.15:1	27.2:1.07:1	NULL

Moreover, the sensor scheduling algorithm that needs to scale the clock frequency or task execution time most often is UDSA, the second most often is BFSA, and the least often is EDF-BSA.

We compare the energy consumed by task executions using DVFS-PTEA(b) and ETS-PTEA(c) with that consumed by continuous task execution(a) in the cases of EDF-BSA(d), UDSA(e) and BFSA(f), and show the ratio of the energy consumption of each case in table 4.2. We calculate the consumed energy by using the formula $E = P * t$, where E, P and t represent the consumed energy, consumed power, and task execution time, respectively. Moreover, we use the formula $P = C * V^2 * f$ [7] to calculate the consumed power. Here, C is the efficient switch capacitance, V is supply voltage, and f is clock frequency. Since the clock frequency varies along with the supply voltage linearly, $f = \alpha * V$ holds, where α is a constant. Therefore, $E = C * f^3 * t * \alpha$ holds, we can calculate and compare the energies consumed by the task in EDF-BSA, UDSA, and BFSA by using this formula. In the evaluation, since the data acquisition frequency of environmental sensors in our system is around 1 second to 1 minute, we assumed in $\Delta t = 10s$, the tasks are periodically invoked 100 times. In each period, a task is executed in $1ms$ and in sleep in $99ms$. Moreover, when the clock frequency is scaled to $n * f$ or the task execution time is scaled to $n * t_c$, the current frequency or task execution time should last for 10 rounds and after that scale to f or t_c to make sure there is enough time to activate or deactivate n sensors at the same time. Table 3.2 shows the results of the evaluation of DVFS-PTEA and ETS-PTEA (the measurement time is in second). It is clear that ETS-PTEA is the most energy efficient solution for scheduling tasks and continuous task execution is the least efficient.

Chapter 5

Simulations

We developed a simulator to evaluate the battery life managed with each of the three algorithms. First, we give a description of our simulator. Then, we show simulation results about the three algorithms.

5.1 Characteristics of the Simulator

The simulator was made to compare battery lives with SA, where SA stands for simultaneous activation and activates all of sensors at the same time. As we mentioned in chapter 3, the time usage of T_{DT} and T_{Sleep} were not taken into consideration in the simulations. The input includes the sensors' character set $< L_i, d_i, k_i >$, number of sensors N, and the discharge current of each sensor. The output returns the remaining battery charge and remaining battery voltage for each algorithm over time. In the simulation, we measured the battery charge and voltage in order to see how long we can extend battery life. We assumed a button battery with an initial battery charge of 620 mAh and battery voltage of 3 V. The operation voltage of the battery was 2.5 V.

When a sensor is chosen, the power consumed by the sensor is fixed. For example, suppose three sensors have been scheduled. The total power consumption is the same in one period, no matter how the sensors were scheduled.

Therefore, we calculated the power consumption in one period by using

$$C_{drop} = \sum I_i k_i$$

where I_i is the operating current of sensor σ_i and k_i is the sensor's data acquisition time. The remaining battery charge at time t can be calculated in

$$C_r(t) = C_r(t - T_p) - C_{drop}$$

where T_p is the time of one period and $C_r(t-T_P)$ is the remaining battery charge one period before. This formulation means that the remaining battery charge at time t is equivalent to the battery charge one period before minus the power consumed by all sensors in one period. Moreover, we approximated

$$U_{drop} = \sum d_{t_k}^2 / w$$

to calculate the voltage drop in one period, where w is a constant obtained by the experiment described in chapter 3. We explain why we use this formula. Since the voltage drop is difficult to formulate for it depends on many chemical and physical factors and there is no standard for calculating the voltage drops of different systems. Instead, we used the voltage drop rules obtained in the experiment described in chapter 3 where voltage drop is in proportion to the square of d. Thus, the formula of voltage drop used in the simulator is a approximate formula. Using the above U_{drop}, we calculated the remaining battery voltage by using

$$U_r(t) = U_r(t-T_p) - U_{drop}$$

where $U_r(t)$ and $U_r(t-T_P)$ are the remaining battery voltage at time t and the remaining battery voltage one period before, respectively.

5.2 Simulation Results

We will show some simulation results of three kinds of d value in order to evaluate the effectiveness of our three algorithms on SA. In SA, the power consumption is the largest. The three kinds of d include the increasing order of d, decreasing order of d and random order of d. Since the operating current of sensors scales from less than $1\mu A$ to near $100\mu A$, we took I to be the operating current shown in Table 5.1 through Table 5.7.

We show six simulation results with the same sensor parameter set but different d which is different in orders.

Table 5.1: Sensor parameter set in simulation 1

	σ_0	σ_1	σ_2
L	5	7	10
k	4	5	5
d	1	2	3
$I(\mu A)$	5	5	5

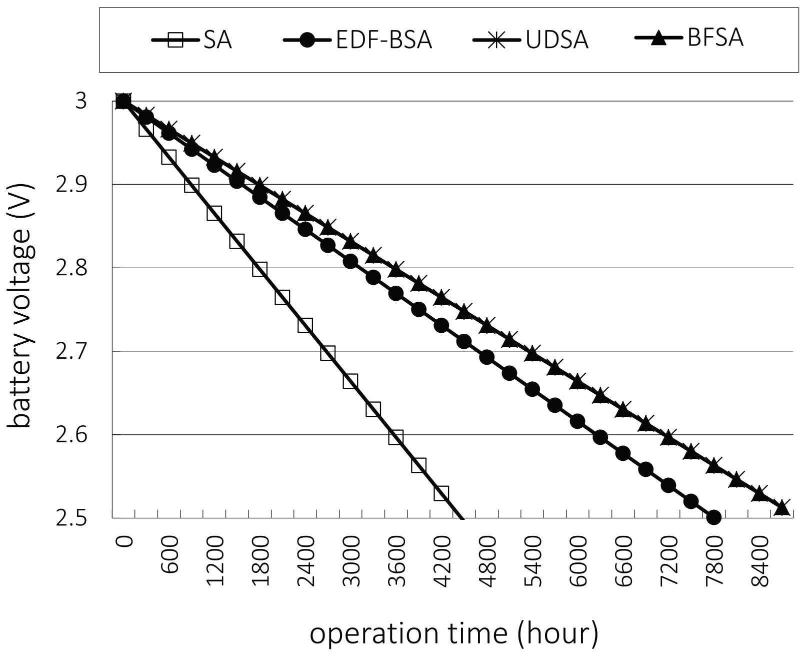


Figure 5.1: Sensor scheduling simulation 1

In the first simulation, we use the sensor parameter set shown in Table 5.1. Figure 5.1shows simulation results of the three algorithms, where *d* is in an increasing order. In this case, when the battery voltage dropped to the operating voltage, EDF-BSA, UDSA and BFSA can extend battery life by about 1.7 times, 2.1 times and 2.1 times, respectively by comparing the 4200 hours sensor lives of SA. Moreover, we observed UDSA has the same effectiveness as BFSA in this increasing order *d* case, and EDF-BSA has a worse effectiveness comparing with UDSA and BFSA.

Table 5.2: Sensor parameter set in simulation 2

	σ_0	σ_1	σ_2
L	5	7	10
k	5	5	4
d	3	2	1
$I(\mu A)$	5	5	5

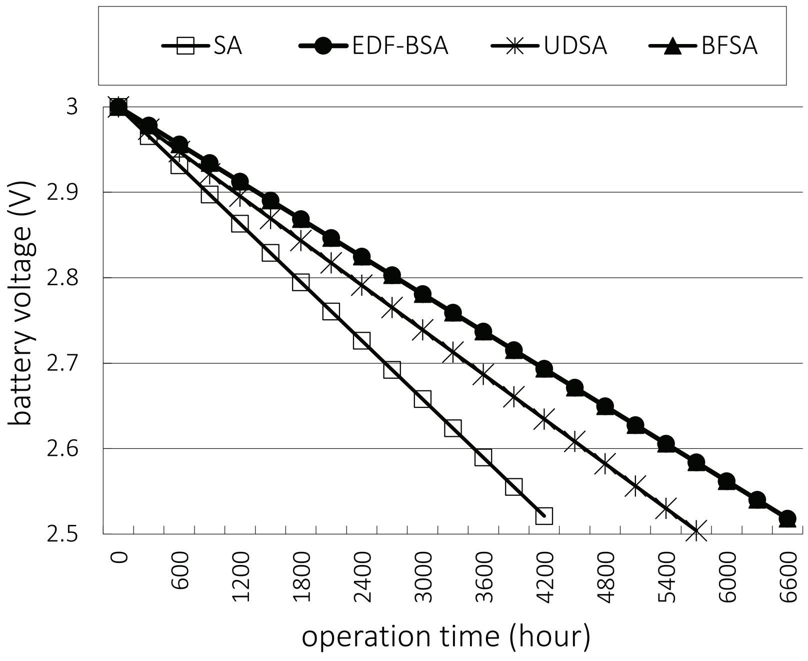


Figure 5.2: Sensor scheduling simulation 2

In the second simulation, *d* has a decreasing order shown in Table 5.2. A simulation result (Fig. 5.2) shows that EDF-BSA, UDSA and BFSA can extend battery life by about 1.6 times, 1.4 times and 1.6 times, respectively by comparing with the 4200 hours sensor lives of SA. In this simulation, BFSA has the same effectiveness as EDF-BSA since *d* is in decreasing order and there is no LPCT to be filled in, thus, line 02 through line 15 in BFSA are not executed. In this case, BFSA is the same as EDF-BSA. Moreover, In UDSA, it causes a “border-crossing” phenomenon in order to meet each sensor’s deadline, therefore, UDSA has a worse effectiveness than BFSA and EDF-BSA.

Table 5.3: Sensor parameter set in simulation 3

	σ_0	σ_1	σ_2
L	5	7	10
k	4	5	5
d	1	3	2
$I(\mu A)$	5	5	5

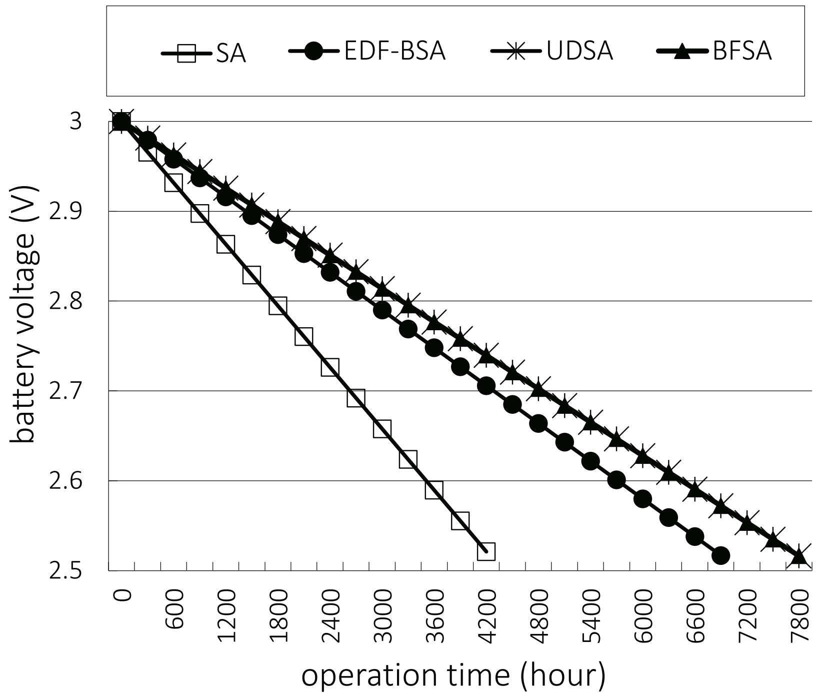


Figure 5.3: Sensor scheduling simulation 3

From the third to the sixth simulation, we conduct simulations with random order of d. In the third simulation, the sensor parameter set with a random order of d is shown in Table. 5.3, and simulation result is shown in Fig. 5.3. It shows that EDF-BSA, UDSA and BFSA can extend battery life by about 1.6 times, 1.9 times and 1.9 times, respectively by comparing with the battery life of SA. UDSA has the same effect as BFSA in this simulation, this is because UDSA has the same schedule as BFSA, and we say this is the best case of UDSA.

Table 5.4: Sensor parameter set in simulation 4

	σ_0	σ_1	σ_2
L	5	7	10
k	5	4	5
d	2	1	3
$I(\mu A)$	5	5	5

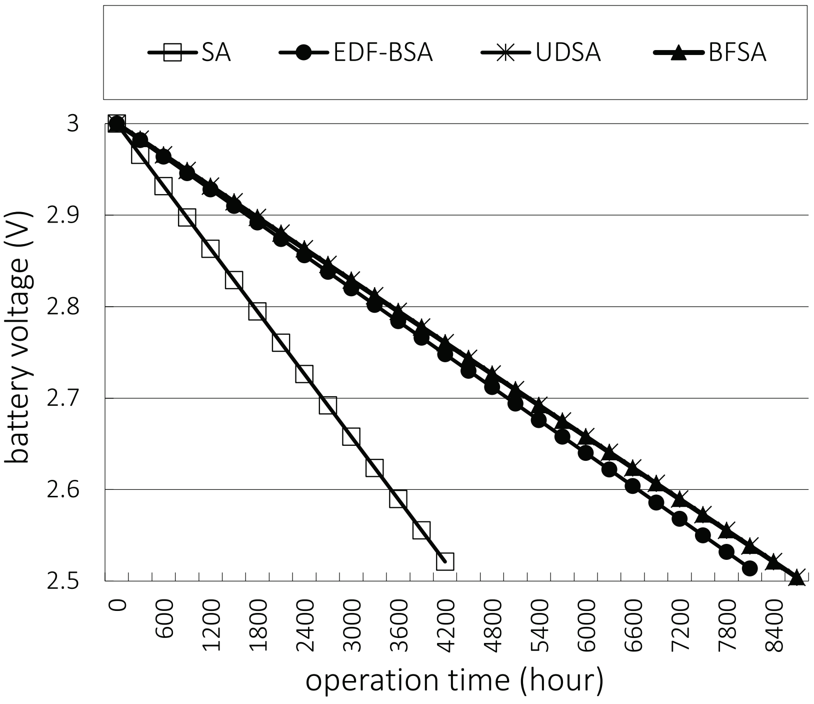


Figure 5.4: Sensor scheduling simulation 4

The fourth simulation uses the sensor parameter set shown in Table. 5.4. The simulation result (Fig. 5.4) shows that the battery life can be extended in about 1.9 times, 2.1 times and 2.1 times in EDF-BSA, UDSA and BFSA, respectively. UDSA has its best case, and EDF-BSA performs well since its effect is close to BFSA.

Table 5.5: Sensor parameter set in simulation 5

	σ_0	σ_1	σ_2
L	5	7	10
k	5	4	5
d	3	1	2
$I(\mu A)$	5	5	5

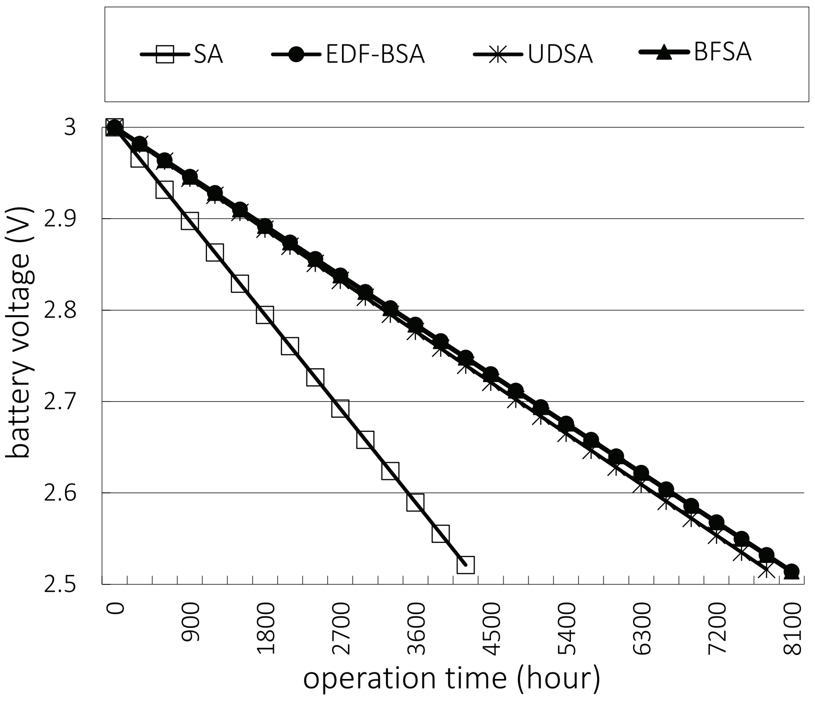


Figure 5.5: Sensor scheduling simulation 5

The fifth simulation uses the sensor parameter set shown in Table. 5.5. The simulation result (Fig. 5.5) shows that the battery life can be extended in about 1.9 times, 1.8 times and 1.9 times in EDF-BSA, UDSA and BFSA, respectively. BFSA performs as good as EDF-BSA, and UDSA performs worse than EDF-BSA and BFSA in this simulation since the value of d in σ_0 is large enough that there is no LPCT can be filled in for σ_1 and σ_2.

Table 5.6: Sensor parameter set in simulation 6

	σ_0	σ_1	σ_2
L	5	7	10
k	5	5	4
d	2	3	1
$I(\mu A)$	5	5	5

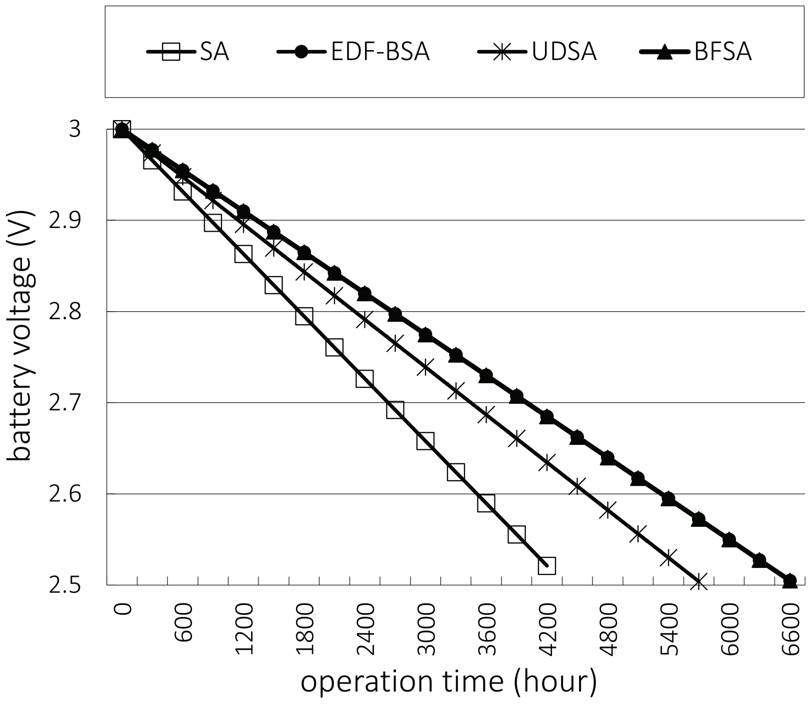


Figure 5.6: Sensor scheduling simulation 6

In the sixth simulation, we use the sensor parameter set shown in Table. 5.6. Fig. 5.6 shows the simulation result that the battery life can be extended in about 1.6 times, 1.4 times and 1.6 times in EDF-BSA, UDSA and BFSA, respectively. Since the same reason in simulation five, BFSA has the same effect as EDF-BSA.

The above simulations show the results of all the proper sensor parameter set of different order of d in a sensor node with three sensors. We show the overall simulation results in Fig. 5.7, and we describe our observations.

Firstly, we observed that when d is in the random order, the effectiveness of UDSA is between the effectiveness of which d is in the increasing order and the decreasing order. When d is in the decreasing order and the random order, the value of $\sum_{\sigma_l \in S(T_a^{\sigma_m})} d_l$ (line 10 in UDSA) tends to be larger than which d is in the increasing order since the earlier sensors have larger d than the later. This causes a "border-crossing" more often than the

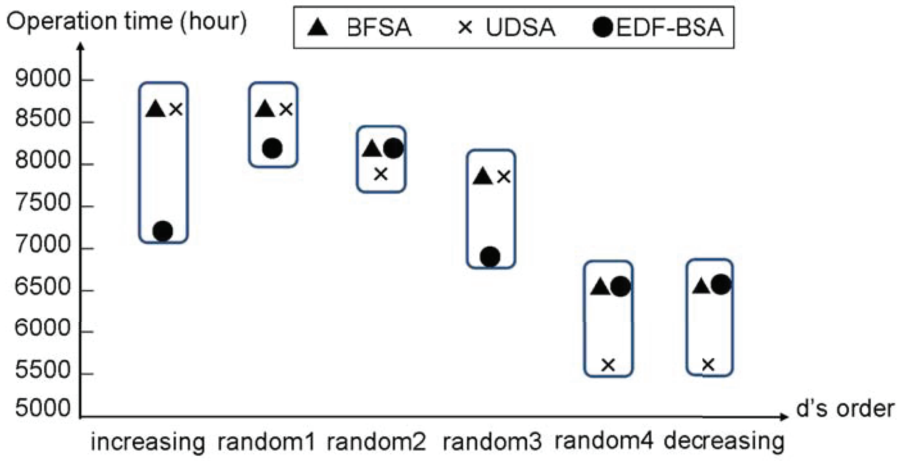


Figure 5.7: Overall simulation results

Table 5.7: Sensor parameter set in simulation 7

	σ_0	σ_1	σ_2	σ_3	σ_4
L	5	7	8	10	12
k	4	5	3	5	5
d	1	1	2	2.5	2.5
$I(\mu A)$	2	3	7	6	8

case of the increasing order of d because $\sum_{\sigma_l \in S(T_a^{\sigma_m})} d_l + d_i \leq d_{over\_ave}$ holds in few cases at line 10 in UDSA. The more "border-crossing" UDSA has, the worse its effectiveness is. Secondly, we observed that in the random order of d, BFSA's effectiveness has the same tendency as UDSA has, because LPCT in BFSA is smaller in the case of the random order of d than the case of the increasing order of d. Lastly, we observed that the effectiveness of EDF-BSA has no rules in accordance to d's order since EDF-BSA depends on each sensor's deadline not values of d.

Moreover, in others simulations, we observed that our algorithms can effectively extend battery lives more than 3 times at most. For example, by using the sensor parameter set shown in Table 5.7, Fig. 5.8 shows that BFSA can extend battery life about 3.27 times.

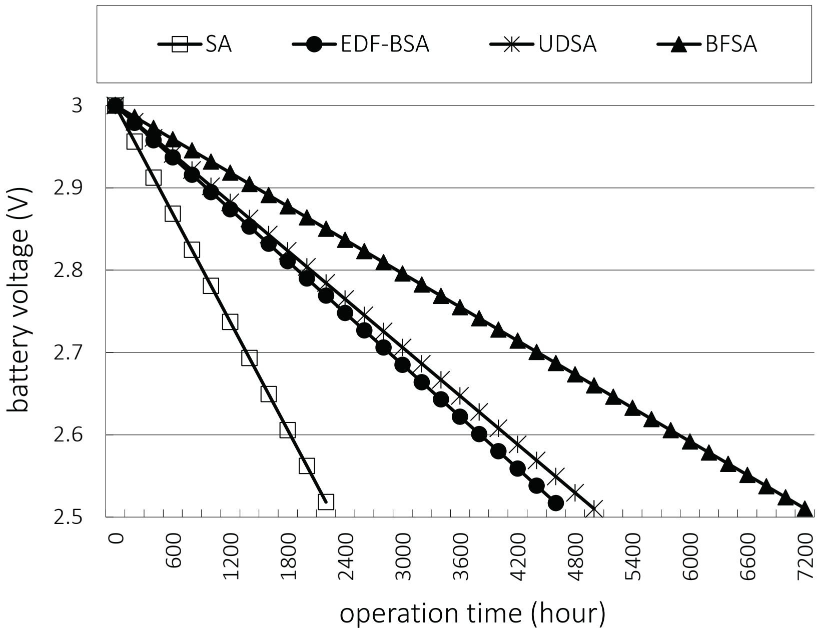


Figure 5.8: Sensor scheduling simulation 7

Part II

Energy Efficient Wireless Communication

Chapter 6

Related Works

In this chapter, we introduce several issues related to wireless communications, and show some of the related works which attempted to solve those issues. As we mentioned in section 1, in a battery powered WSNs, electric power is mainly consumed by wireless communications through sensors in the nodes and by CPUs running programs to control the sensor nodes. Since wireless communication is the dominant power consumer [25] in sensor nodes, consuming up to approximately 80% of their power [26], the wireless communication power should be minimized to improve their energy efficiency. In this thesis, we focused on three topics which need to be solve in order to reduce energy consumed by wireless communications in a collaborative manner.

As soon as a sensor node obtains sensing data, it should transmit these data to the base station directly or via other nodes by using a specific wireless communication technology such as ZigBee. In wireless communications, energy consumption is mainly affected by communication distance and the size of transmitted data, especially, communication distance is the crucial component since the energy consumption equation refers to

$$E_d = ad^{\alpha} + b \tag{6.1}$$

where d is communication distance, a and b are positive constants, and $2 \leq \alpha \leq 6$ due to different path loss model a WSN utilized [27][28]. For example, $\alpha = 2$ in free space model, $\alpha = 4$ in two-ray model, and $\alpha = 2$ to 6 in shadowing model. Therefore, it is essential to utilize an energy aware data transmission routing in order to avoid the extra energy consumption caused by long communication distance. Generally, direct data transmission may consume more energy if the distances between sensor nodes and base station are too long [29]. Therefore, if the communication distance is long, a solution for avoiding high energy consumption by applying a short communication distance route is required.

Another problem existing in WSNs is the energy hole phenomenon [27], in which it is known that a node terminates quicker than the other nodes if it relays data more often than the others. This is because multi-hop data transmission is usually used in a data communication route in order to reduce communication distance. And nodes which play the role of internal nodes (data-relaying nodes) consume more energy than the others since they relays data from more nodes than child nodes in the topology. Therefore, the energy hole problem may produce non-uniform energy consumptions in the WSN, as a result, internal nodes terminate quicker than other child nodes, and the child nodes have to find another internal node to relay their data, which bring a short lifetime for the entire network. In view of extending the lifetime by judicious communication route selection, it is desirable to solve the energy hole problem.

Moreover, a fault tolerance problem that do not directly affect the energy consumption of wireless communications but can truly affect the entire network lifetime exists in a WSN. In many applications, a WSN is deployed in harsh environments such as in a forest or around a volcano, where sensor nodes may be damaged. In addition, sensor nodes may also fail due to permanently interrupted wireless communication or hardware crash down, such as breakdown in on-board electronics. In this situation, network connectivity will be damaged, and left part of the network uncovered. Therefore, it is important that a WSN is able to tolerate such faults.

In the following of this chapter, we introduce some related works which attempted to solve these problems respectively.

6.1 Energy Efficient Routing Approaches

6.1.1 Energy Efficient Routing Algorithms

Energy consumed in wireless communication is affected by two main factors; the first is the amount of data and the second is the communication distance. Apparently from equation 6.1, a communication route which can shorten the communication distance is preferred. Generally, there are two ways to relay data from a source node to a destination node or a base station, they are direct data transmission and multi-hop data transmission. Since direct data transmission consumes more energy due to a longer communication distance, multi-hop data transmission is usually used in wireless communications. In an energy aware multi-hop WSN which all the data of sensor nodes are required to be sent to a base station, sensor nodes should be selected to a route by an energy efficient method according to equation 6.1. In this section, we introduce some famous energy efficient routing algorithms.

Some work on minimizing the amount of data has had to focus on how to gather data effectively such as the sensor protocol for information via negotiation (SPIN) [30], which takes into consideration the data negotiation between nodes to eliminate redundant data and COUGAR [31], which uses declarative queries to abstract query processing from the network layer functions and utilizes in-network data aggregation to conserve energy. However, this thesis focuses on reducing the communication distance based on network architecture to achieve energy efficiency, and we introduce some energy-efficient communication protocols that focus on shortening the communication distance.

LEACH [29] is known as a famous energy-efficient protocol for forming clusters that minimizes the transmission routes and amounts of data. A small number of sensor nodes in the sensor network in LEACH are divided into clusters where individual sensor nodes are selected as cluster heads in a self-organizing manner. The cluster heads collect the sensing data from each node in the cluster and aggregate them into one single data group; after the data are aggregated, they are sent from the cluster heads to the base station. However, Heinzelman et al. found that the nodes that were selected as cluster heads quickly expired, which terminated the lifetimes of their clusters. Thus, LEACH used randomization to rotate the cluster heads to prevent cluster heads from quickly terminating and its wireless communication cost could be reduced as much as eight times compared with direct transmission.

Lindsay and Raghavendra proposed further improvements to LEACH in [32] by taking advantage of each sensor node only communicating with its closest neighbors, and only one designated sensor node sending aggregated data to the base station in each round. This approach could be used to distribute power consumption evenly among sensor nodes in a WSN. First, PEGASIS constructed a chain to determine which node should be transmitting or receiving data from which node. Since nodes on the chain could not be revisited, the distance to their neighbors increased. Moreover, the chain should be reconstructed in the same manner when a node terminates. PEGASIS gathers data in each round after the chain has been constructed. Each node receives data from one neighboring node, aggregates these with its own sensing data, and transmits the sensing data to other neighboring nodes on the chain. Finally, a leader node receives all the sensing data from the others, aggregates these data with its own data, and transmits one message to the base station. The simulation results indicated that PEGASIS outperformed LEACH by about one to three times.

6.1.2 Minimum Spanning Tree Routing

Another branch of researches attempted to minimize communication distance by using graph-theoretic structures. Considering a WSN as a graph $G = (V,E)$, where V and E are vertex set and edge set, respectively; hence, sensor nodes and communication link between nodes can be considered as vertexes and edges, respectively. Therefore, forming a data communication route has the same meaning of generating a spanning tree in a graph since a spanning tree may include all vertexes in a graph. Moreover, a minimum spanning tree (MST), which contains all the nodes and whose sum of edge lengths is minimized, is beneficial for wireless communication routes requiring minimized communication distance. It can be constructed using the algorithms made by Prim, Kruskal and Dijkstra [33].

```
MST-PRIM(G,w,r)
01 for each $u \in Q$
02     do $key[u] \leftarrow \infty$
03        $\pi[u] \leftarrow NIL$
04 $key[r] \leftarrow 0$
05 Q ← V(G)
06 while $Q \neq 0$
07     do u ← EXTRACT-MIN(Q)
08        for each $v \in Adj[u]$
09            do if $v \in Q$ and $w(v,u) < key[v]$
10                then $\pi[v] \leftarrow u$
11                     $key[v] \leftarrow w(v,u)$
```

In Prim's algorithm [33], the spanning tree starts from an arbitrary root vertex r and grows until the tree spans all the vertexes in V. At each tree growing step, a light edge is added to tree A that connects A to an isolated vertex of $G_A = (V,A)$. This rule adds only edges that are safe for A (a safe edge for A is an edge (u,v) not in A). Prim's algorithm is shown in $MST-PRIM(G,w,r)$. In the pseudo code, the connected graph G and the root r of a minimum spanning tree to be grown are inputs to the algorithm. During execution of the algorithm, all vertexes that are not in the tree reside in a min-priority queue Q based on a *key* field. For each vertex v, $key[v]$ is the minimum weight of any edge connecting v to a vertex in the tree. $key[v] = \infty$ means there is no such edge. Field $\pi[v]$ names the parent of v in the tree. During the algorithm, the set A is $A = \{(v,\pi[v] : v \in V - \{r\} - Q\}$. When the algorithm terminates, queue Q is empty, the minimum spanning tree A for G is $A = \{(v,\pi[v] : v \in V - \{r\}\}$. We show how Prim's algorithm works. Lines 1-5 set the key of each vertex to ∞ (except for the root, whose key is set to 0 so that it will be the first vertex processed), set the parent of each vertex to NIL, and initialize the min-priority

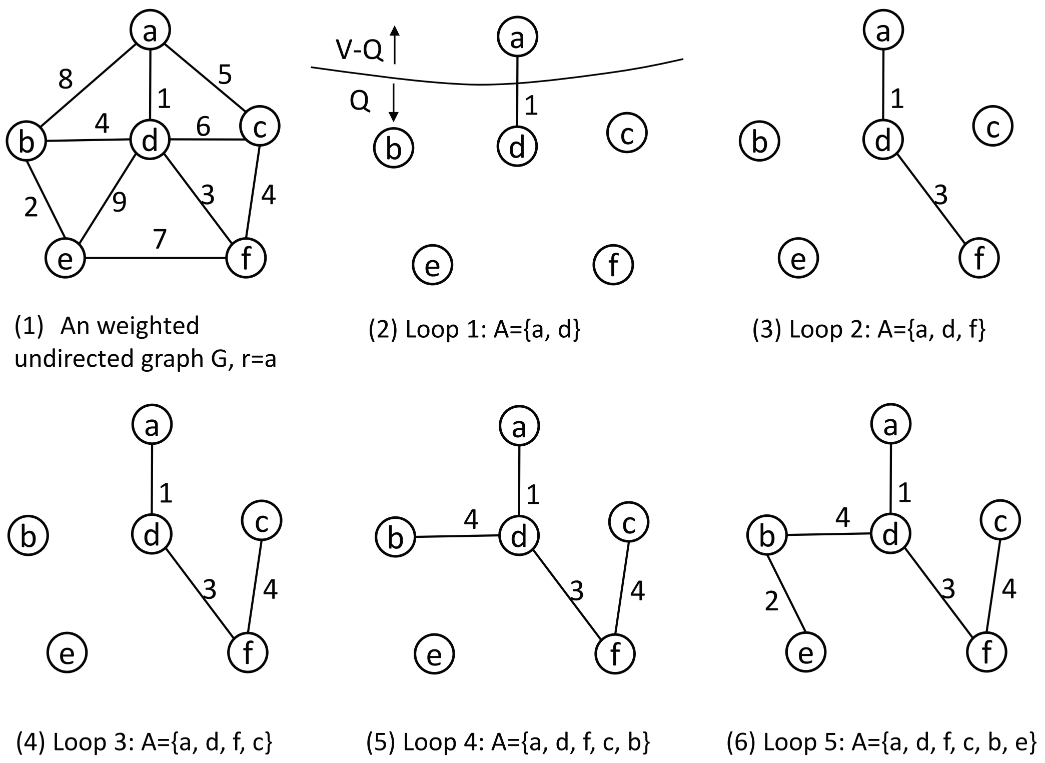

(1) An weighted undirected graph G, r=a
(2) Loop 1: A={a, d}
(3) Loop 2: A={a, d, f}
(4) Loop 3: A={a, d, f, c}
(5) Loop 4: A={a, d, f, c, b}
(6) Loop 5: A={a, d, f, c, b, e}
V-Q
Q
a
b
c
d
e
f
8
5
1
4
6
2
9
3
4
7

Figure 6.1: Execution of Prim's algorithm

queue Q to contain all the vertexes. The algorithm maintains the following three-part loop invariant:

Prior to each iteration of the **while** loop of lines 6-11,

1. $A = \{(v, \pi[v] : v \in V - \{r\} - Q\}$

2. The vertexes already placed into the minimum spanning tree are those in $V - Q$

3. For all vertexes $v \in Q$, if $\pi[v] \neq NIL$, then $key[v] < \infty$ and $key[v]$ is the weight of a light edge $(v, \pi[v])$ connecting v to some vertex already placed into the minimum spanning tree.

Line 7 identifies a vertex $u \in Q$ incident on a light edge crossing the cut $(V - Q, Q)$ (with the exception of the first iteration, in which $u = r$ due to line 4). Removing u from the set Q adds it to the set $V - Q$ of vertexes in the tree, thus adding $(u, \pi[u])$ to A. The for loop of lines 8-11 updates the *key* and π fields of every vertex v adjacent to u but not in the tree. The updating maintains the third part of the invariant. The complexity of Prim's algorithm is $O(|E| \lg |V|)$.

Figure 6.1 shows the execution of Prim's algorithm for a six-vertex graph G. Figure 6.1(1) shows an weighted undirected graph, we assume the root of minimum spanning tree A is node a. According to Prim's algorithm, the lightest edge crossing the cut $(V - Q, Q)$

should be selected to the tree. So in loop 1 in Fig. 6.1(2), the light edge crossing the cut are edges (a,b), (a,c) and (a,d), and the lightest edge among them is (a,d), hence, (a,d) is selected to tree A. Same procedures are applied in loop 2 to loop 5 in Fig. 6.1(3)-(6) until the minimum spanning tree is formed in Fig. 6.1(6).

Other minimum spanning tree algorithms include Kruskal's algorithm and Dijkstra's algorithm. Kruskal's algorithm [33] finds a safe edge to add to the growing tree by finding all edges that connect any two subtrees in the tree: an edge (u,v) of least weight. It is different from Prim's algorithm in that its tree growing procedure starts from an edge with least weight in the graph, contrarily, an arbitrary root in prim's algorithm. The complexity of Kruskal's algorithm is $O(|E|\lg|V|)$. Dijkstra's algorithm [33] solves the single-source shortest-paths problem on a weighted directed graph. Dijkstra's algorithm maintains a set S of vertexes whose final shortest-path weights from the source s have already been determined. The algorithm repeatedly selects the vertex $u \in V-S$ with the minimum shortest-path estimate, adds u to S, and relaxes all edges leaving u. The complexity of Dijkstra's algorithm is $O(|E|+|V|\lg|V|)$.

6.2 Energy Hole Aware Approaches

As we mentioned before in this chapter, energy hole problem is known as the uneven energy depletion phenomenon in WSNs, and it is critical to be solved. In [27], the authors investigated theoretical aspects of the energy hole problem. The author assume an energy consumption model of $E = d^{\alpha} + c$, where d ($d \leq t_x$) is the transmission distance, $\alpha \geq 2$ is the power attenuation, c is a technology-dependent positive constant, and t_x is the minimum transmission range of sensors. The authors found that, in a sink based WSN, for $\alpha > 2$, all sensors whose distance to the sink is $min\{t_x, (\frac{2c}{\alpha-2})^{\frac{1}{\alpha}}\}$ should transmit directly to the sink. Moreover, when $\alpha > 2$, non-uniform energy consumption can be prevented by judicious system designs, such as placing data sinks at strategic locations and adjusting the network radius around the sinks. The authors also stated that in such a system the energy expenditure is balanced across the network, which means that judicious system design can result in uniform energy consumption among the sensor nodes. Another solution, reported in [34], was to attempt to mandate the sinks to move around in such a way that some load balancing is obtained across the deployment area. In [35], the authors proposed another solution in which the number of nodes in each corona is regulated and the ratio between the node densities in two adjacent coronas is derived to mitigate the energy hole problem.

Several recent studies investigated energy efficient routing and proposed methods for solving the energy hole problem [36] [37] [38]. Liu surveyed various problems in WSNs including the energy hole problem and routing problems and introduced representative methods for solving them [36]. Tanessakulwattana et al. proposed improving energy efficiency and solving the energy hole problem by enabling each node to send packets to neighbor nodes in different proportions so as to more evenly distribute energy consumption among the nodes [37]. In [38], Jun et al. proposed solving the energy hole problem by enabling the nodes with more residual energy to directly communicate with the sink by utilizing cooperative transmission (CT) for range extension. Our work is different in that we propose methods not only to solve the energy hole problem but also to reduce total energy consumption and to adapt the WSN with random node distribution which are not mentioned in the above studies.

6.3 Fault Tolerance Approaches

Some researchers focused on enhancing the fault tolerance of data communications. There are two methods for handling the fault tolerance problem [39]: (1) proactive approaches that attempt to provision resources in the network topology to tolerate node failures, and (2) reactive approaches that attempt to tolerate node failures through real time restoration of lost connectivity. In this thesis, we focused on the proactive approach. There are two variants to the proactive approach. In the first, fault tolerant topologies are formed at the time of deployment. We did not consider this approach because WSN nodes in infrastructures such as bridges should be deployed at fixed positions. In the second, we form a k-vertex connected (referred to as k-connectivity) WSN, or designate backup nodes for critical nodes in the network. A k-connectivity network has a $k-1$ fault tolerance, which means it can tolerate the failure of up to $k-1$ nodes [40].

An algorithm aimed at reducing communication energy while preserving network connectivity was proposed by Li and Hou [41]. They presented a centralized algorithm called the fault-tolerant global spanning subgraph ($FGSS_k$), which was based on Kruskal's algorithm. Different components are iteratively merged until only one k-connected component remains. A localized algorithm called the fault-tolerant local spanning subgraph ($FLSS_k$) was based on $FGSS_k$, and proposed to control the topology. The authors showed that $FGSS_k$ and $FLSS_k$ can preserve k-connectivity and minimize the maximum transmission power of the network. Moreover, $FLSS_k$ maintains bi-directionality for all links in the topology. Li et al. investigated the minimum transmission range that guarantees k-connectivity with a high probability [42]. They also presented a localized method for controlling the network topology that was based on the Yao structure and preserves k-connectivity. Jorgic et al. proposed an alternative fault tolerance method that uses localized algorithms to detect the critical nodes and links [43]. Other works have attempted to generate *k*-connectivity, including [44, 45, 46].

Other proactive approaches designate spares for critical nodes, which act as cut vertexes in the network topology. The simplest way to achieve this is to place redundant nodes that accompany the critical nodes. However, if all the nodes serve the application, the spare nodes must be selected from the active nodes. In [47], Vaidya and Younis proposed NORAS to select spare nodes within the 2-hop neighborhood of a failed critical node. If non exist, the search widens to include more distant nodes. When a critical node fails and is detected, the designated spare nodes travel to replace the critical node, or a series of cascaded relocations on the shortest route between the critical and selected spare nodes are triggered to split the travelling distance. In [48], the authors proposed PADRA, which identifies a connected dominating set for the WSN. PADRA designates a failure handler to each critical node in case it fails. An ideal handler will be a dominatee neighbor of the critical node that can simply replace the critical node. In our approach, we balanced the energy efficiency, energy hole problem, and fault tolerance to obtain reliable energy efficient wireless communications.

Chapter 7

Energy Hole Aware Energy Efficient Routing Algorithms

7.1 Preliminaries and problem Definitions

In this thesis, we assume that in a 100 meter radius circular network, a random number of homogeneous sensor nodes were uniformly distributed in random positions and the base station is located in the center of the circular network as shown in Fig. 7.1. Moreover, the node positions and the distances between nodes are known to the base station by a direct message sending from nodes after they are distributed. Since the maximum communication distance of sensor nodes that use ZigBee device is 100 m [49], this topology ensures that each node in the WSN can reach the base station. Furthermore, we assume that the base station is always powered by a stable power source and therefore the energy consumed by the base station is not included in the wireless communication. We use a free space radio propagation model to calculate the transmitting and receiving costs for a k-bit message at a distance of d as:

Transmitting:

$$E_{T_x}(k,d) = E_{elec} * k + \varepsilon_{amp} * k * d^2 \tag{7.1}$$

Receiving:

$$E_{R_x}(k) = E_{elec} * k \tag{7.2}$$

which are the same as those for LEACH [29], where $E_{elec} = 50nJ/bit$ to run the transmitter or receiver circuitry and $\varepsilon_{amp} = 100pJ/bit/m^2$ to run the transmit amplifier. Since we only focus on extending the lifetime of WSNs by making efficient routing algorithms in wireless communications, we ignore other energy consumptions such as those in CPUs and sensors. Additionally, we assume that each node has 2000 bits of data to send, as in [29] and [32], and that the sending data size is always the same in each node. Therefore,

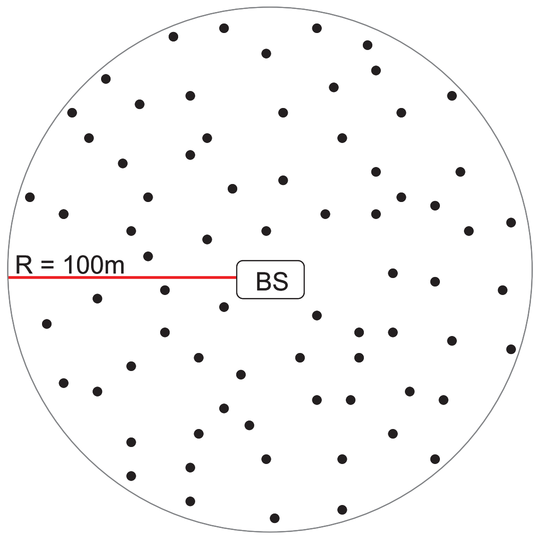


Figure 7.1: Wireless sensor network topology.

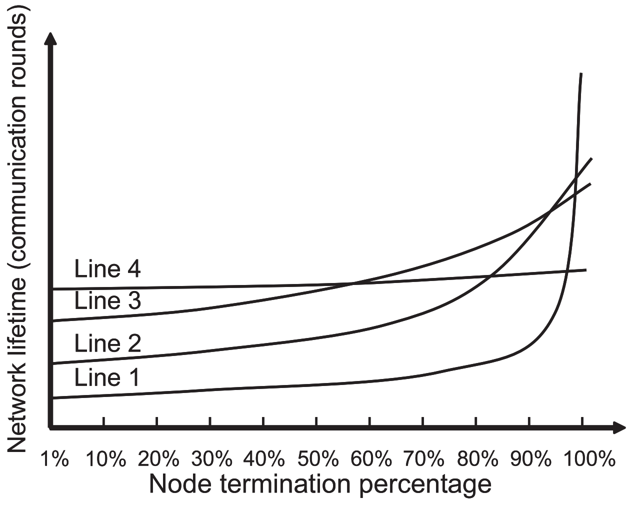


Figure 7.2: Algorithms' performance for different WSN lifetime definitions.

the total energy consumption of node x is $E(x) = E_{T_x}(k,d) + E_{R_x}(k)$ if x is an internal node and $E(x) = E_{T_x}(k,d)$ if x is a leaf node. The energy consumption of the entire network is $E_{sum} = \sum E(x)$.

In this section we will show two problems needing to be solved. First, we define network lifetime and node lifetime before showing the two problems. In a battery powered WSN, the network lifetime is defined as the number of communication rounds till $\alpha\%$ of sensor nodes terminate, where α is specified by the system designer [50]. The node lifetime is defined as the number of communication rounds from the first sensing to the last sensing until the initial battery energy of the node is completely consumed. To extend the lifetime of networks, we should consider cases with different values of α in the network lifetime definition and choose different strategies based on the network lifetime definition. If a high sensor node termination percentage is acceptable in the definition, it means that the WSN can work even if only a small amount of sensor nodes remain working. On the other hand, if a low node termination percentage is required, it means that most of the sensor nodes must remain working to provide the WSN functions. Figure 7.2 shows the network lifetime images of four different wireless communication algorithms as depicted by four different lines. These were obtained in our preliminary simulation experiences, where the x and y axes show the node termination percentage and network lifetime, respectively. Line 1 shows the lifetime image of direct data transmission where data are directly transmitted from each node to the base station. Line 2 shows the lifetime image obtained by using a better algorithm, which could be LEACH, PEGASIS, or PRIM. It can be seen that most of the time, the performance shown by line 2 is better than that shown by line 1. This indicates that when attempts are made to increase the lifetime of the short lifetime nodes, the lifetime of the long lifetime nodes decreases. This is due to new data communication relations between nodes being established as a result of the network routing reconstruction required for increasing the lifetime of the short lifetime nodes. Network routing reconstruction is defined as altering the data transmission path of the nodes in the network for the purpose of extending the node lifetime and network lifetime. We attempt to create new wireless communication routing algorithms that can obtain network lifetime images like those shown by line 3 and line 4 to meet the requirements of different network lifetime definitions. However, network routing reconstruction is required to create a uniform WSN lifetime such as that shown by line 4, i.e., to solve the energy hole problem. This is because a lifetime image like line 4 can only be obtained by network routing reconstruction since, as demonstrated in [27], there is no way to avoid all energy holes by using only one route when applying a wireless communication model such as the one we use here (equations 7.1 and 7.2). On the other hand, network routing reconstruction is not required in creating a lifetime image like line 3 since such an image can be obtained by judicious communication route design.

First, we consider how to create line 3 to adapt to a situation where network lifetime is defined by a high node termination percentage without network routing reconstruction. In order to extend the lifetime of networks, we should minimize the energy consumption of sensor nodes and avoid energy holes by using only one energy efficient route. The method is to uniformly distribute the energy consumption of sensor nodes as much as possible by minimizing the node energy consumed not only by sending but also by relaying data. Taking the reduction of energy consumption for relaying data into account can reduce the receiving energy of the energy hole nodes to solve the energy hole problem. Furthermore, it can ensure that new energy holes will not be produced by the rise in energy consumption of other nodes.

Second, we consider how to create line 4 to adapt to a situation where the network lifetime is defined by a low node termination percentage with network routing reconstruction. In such WSNs, the node lifetimes are required to be almost the same (i.e., energy holes must be avoided to obtain network lifetime as in line 4) so that the energy of long lifetime nodes will not be wasted and the lifetime of short lifetime nodes will be increased by sacrificing the long lifetime nodes. This solution requires a number of switching routes that can be alternatively utilized with the efficient route to rest the energy hole nodes. Consequently, we define the problems as follows:

1. A problem of finding an efficient communication route that can minimize the energy consumption of sending and relaying data without network routing reconstruction.
2. A problem of finding switching routes that can uniformly distribute the energy consumption of sensor nodes to avoid energy holes with network routing reconstruction.

Next, we present conditions for avoiding energy holes in a WSN. As the energy holes are caused by unbalanced energy consumption among nodes, we focus on the energy holes caused by data communications. Note that conditions at the application program level, such as avoiding large energy consumption by application programs, are beyond the scope of this thesis. Two conditions for avoiding energy holes are given here.

1. Routing construction viewpoint: Energy consumption should be balanced in relaying data. This condition requires that every data-relaying node[1] relays the same number of nodes as much as possible to balance the energy consumed by nodes in relaying data.
2. Node lifetime viewpoint: The unbalanced remaining amount of energy in a node is balanced if the amount of energy is unbalanced. This condition requires to equalize the node lifetimes as much as possible.

7.2 Wireless Communication Route Construction

In this section, we explain how to construct an efficient wireless communication route to solve the first problem. We first introduce a wireless communication graph and then explain the algorithm and its properties.

[1]In a tree route, the data-relaying nodes are referred to as internal nodes, and the nodes from which data are relayed by the data-relaying nodes are referred to as child nodes.

7.2.1 Wireless Communication Graph

In this thesis the initial wireless sensor network, the number of which is N, can be described as an undirected weighted graph $G = < V, E >$. Here, $V = \{0, ..., N-1\}$ is a vertex set indicated by the sensor nodes and E is an edge set showing the data communication of two nodes. An edge in the graph is indicated by $(i, j) \in V$. Since the data sending between two nodes determines the direction of an edge, the wireless sensor network graph becomes a directed graph. We let $dir(i, j)$ denote that the direction of edge (i, j) is from i to j. In order to calculate the energy consumption of data communication between two nodes i and j, we use the energy consumption metrics shown in equations 7.1 and 7.2 to define a nonnegative weight $w(i, j) = E_{T_{(i,j)}} + E_{R_{(i,j)}}$ associated with each edge $(i, j) \in V$ with $dir(i, j)$. Note that the notation $E_{T_{(i,j)}}$ is the transmitting energy of node i and $E_{R_{(i,j)}}$ is the receiving energy of node j. $w(i, j)$ represents the total energy consumed by the wireless communication between nodes i and j. Actually, the value of $w(i, j)$ is determined if the network topology is determined since its value only depends on the distance between nodes i and j.

7.2.2 Energy Hole Aware Energy Efficient Communication Routing Algorithm

In this subsection, we propose an energy hole aware energy efficient communication (EHAEC) routing algorithm to solve the first problem mentioned in Sec. 7.1.

Generally, EHAEC is based on the PRIM algorithm [33], which is an optimal algorithm to find the “minimum spanning tree” for an undirected connected weighted graph. What is different in EHAEC is that we involve a concept of “phony weight” in order to avoid energy holes as much as possible in a tree route. Suppose that edges (y_i, b) with direction of $dir(y_i, b)$ exist in a “minimum spanning tree” T, $y_i \in T$ and $i \in N$. A phony weight $P(x, b)$ for $b \in T$ and $x \in V \backslash T$ is defined as:

$$P(x, b) = \sum_{y_i \in T} E_{R_{(y_i, b)}} \tag{7.3}$$

The direction of edge (x, b) is $dir(x, b)$. The phony weight $P(x, b)$ is added to the $w(x, b)$, where edge (x, b) may connect an unconnected node x to a node b in the tree that has already relayed data from other nodes. The idea of phony weight is to avoid a situation in which one node relays data from a lot of nodes, in order to uniformly distribute the energy consumed by sending and relaying data of all the nodes in the network. However, the phony weight $P(x, b)$ is not actually consumed in edge (x, b) since it has already been consumed by edges (y_i, b). Next, we show an example of the effect of phony weight.

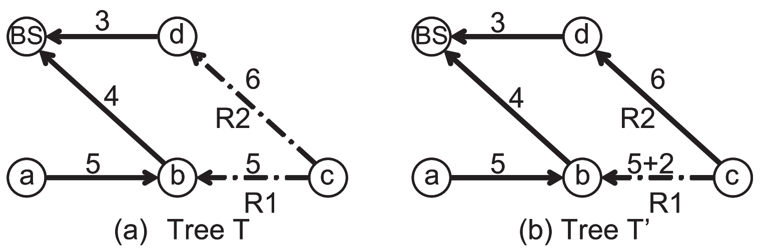


Figure 7.3: Phony weight.

Figure 7.3 shows a procedure of forming T' from T by adding a node c. In Fig. 7.3(a), nodes a, b, and d were added to the tree T. Since we next try to add node c to T, we should choose either route $R1$ or $R2$ to add it [2]. According to the definition of phony weight, since node b has already relayed data from node a, the other edges that may relay data to node b should add the phony weight to themselves. In Fig. 7.3(b), assuming the phony weight is 2, we have $P(c,b) = E_{R_{(a,b)}} = 2$ and $P(c,d) = 0$; therefore, $w(c,b) = 7$ and $w(c,d) = 6$. Due to the updated weight of the edge, route $R2$ is selected since $w(c,d) < w(c,b)$; thus, tree T' is formed.

```
Algorithm 1: EHAEC(G,w,r)
01 Q ← V(G)
02 for each $u \in Q$
03     $key[u] \leftarrow \infty$
04     $\pi[u] \leftarrow NIL$
05 $key[r] \leftarrow 0$
06 while $Q \neq 0$
07     $u \leftarrow$ EXTRACT-MIN(Q)
08     $A \leftarrow \{(u, \pi[u]) : v \in V - r - Q\}$
09     edge direction: $dir(u, \pi[u])$
10     for each $v \in Adj[u]$
11         for each $s \in Adj[\pi[u]] - u$ and $s \in V \backslash A, s \neq r, u \neq r, \pi[u] \neq r$
12             $P(s, \pi[u]) = \sum_{x \in A} E_R(x, \pi[x])$, where $\pi[x] = \pi[u]$
13             $w(s, \pi[u]) = w(s, \pi[u]) + P(s, \pi[u])$
14             if $s \in Q$ and $\pi[s] = \pi[u]$ and $key[s]$ is $w(s, \pi[u])$ that in last round
15                 $key[s] \leftarrow w(s, \pi[u])$
16         if $v \in Q$ and $w(v,u) < key[v]$
17             $\pi[v] \leftarrow u$
18             $key[v] \leftarrow w(v,u)$
19 return $A$
```

EHAEC works in the following way. The required inputs are graph G of the WSN,

[2] Actually, there are other routes to send node c's data to BS, such as edges (c, BS) and (c, a). However, since these edges have large weight, we ignored the two routes in Fig. 7.3

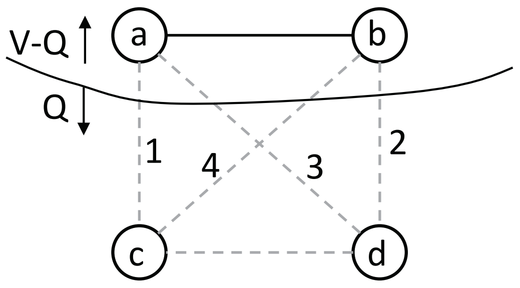


Figure 7.4: Explanation of EXTRACT-MIN(Q).

and root r (the base station) of G. The output returns spanning tree A, which avoids energy holes as much as possible without reconfiguring the network. In EHAEC, all vertexes not in the tree are stored in a priority queue, Q. The weight of each edge are calculated a prior. Weight of an edge (v,u) is defined as $w(v,u) = E_{T_{(v,u)}} + E_{R_{(v,u)}}$. For each vertex v, $key[v]$ is the minimum weight of any edge connecting v to a vertex in the tree. Field $\pi[v]$ indicates the parent of v in the tree. Lines 1-5 of the algorithm are the initializations. While Q is not empty, Line 7 extracts a vertex $u \in Q$ incident on a light edge crossing the cut $(V-Q,Q)$ (with the exception of the first iteration, in which $u = r$ due to line 5)[3], which means it extracts a node u which should be add to the tree in the current loop. Line 8 stores the parent-child vertex set in tree A. Line 09 defines the direction of edge $(u,\pi[u])$. For every vertex v that is adjacent to u but not in the tree (line 10), line 12 defines the phony weight $P(s,\pi[u])$ and line 13 updates the edge weight $w(s,\pi[u])$ if vertex s is adjacent to $Adj[\pi[u]] - u$, and vertexes s, u and $\pi[u]$ are not the root (line 11). Except for root r, if a parent $\pi[u]$ is already connected to a child x, phony weight $P(s,\pi[u])$ is defined to avoid connecting another vertex s to $\pi[u]$ because this may cause an energy hole in vertex $\pi[u]$. Lines 14 and 15 determine if $key[s]$ should be updated. In Lines 16-18, for every vertex v that is adjacent to u but not in the tree (line 10), if edge (v,u) has the smallest weight, the *key* and π fields of every vertex v are updated. The updated *key* contains the vertex which has the minimum weight edge that may be extract in the next loop. Finally, in Line 19, EHAEC returns the spanning tree A. EHAEC has the same complexity as Prim's algorithm, which is $O(|E|\lg|V|)$.

Theorem 1 *EHAEC constructs a communication route that has the least energy consumption in sending and relaying data* [4] *in WSNs.*

[3]We use Fig. 7.4 to explain EXTRACT-MIN(Q). Supposing vertexes a and b are in the tree. Hence, $Q = \{c,d\}$ and $V - Q = \{a,b\}$. The black curve indicates the idea of cut $(V-Q,Q)$. Since EXTRACT-MIN(Q) extracts a light edge crossing the cut, vertex c is extracted because edge (a,c) is the light edge among edges (a,c), (a,d), (b,c) and (b,d).

[4]Nodes that can send and relay data are limited to the internal nodes since a leaf node can only send data.

Proof: The theorem means the following: assuming that tree T with $n(n > 1)$ nodes constructed by EHAEC exists, $x \in T$ is an internal node and $y \in T$ is a leaf or internal node. In the procedure of forming tree T' from T by adding a leaf node $a \in V \backslash T$ (since a is a leaf node, only the energy it consumes for sending is taken into consideration), on the assumption that edge (a,x) has the minimum weight in the current step of forming T', if the initial weight

$$w(a,x) < w(a,y)$$

and

$$w(a,x) + P(a,x) > w(a,y) + P(a,y)$$

where phony weight $P \geq 0$, then node x has the least energy consumption in sending and relaying data in T' compared with node x in other trees. Since the other internal nodes have the same property as node x without loss of generality, we can prove the theorem by only proving the property of node x.

In fact, if

$$w(a,x) < w(a,y)$$

and

$$w(a,x) + P(a,x) > w(a,y) + P(a,y)$$

then node a will be a child of node y instead of being the child of x by taking advantage of the phony weight, $E_{R_{(a,y)}}$, which is the energy consumed by receiving data from node a and will be consumed by edge (a,y) but not edge (a,x). Note that under this condition, adding an edge (a,x) to tree T is equivalent to constructing the minimum spanning tree in PRIM. We prove the theorem by induction.

Base step: For $n = 2$, there are two nodes x and y in tree T. The relation of nodes x and y is $dir(y,x)$ since node x is an internal node, hence,

$$E(x) = E_{R_{(y,x)}}$$

and

$$E(y) = E_{T_{(y,x)}}$$

If node a is added to y in T', then

$$E(x) = E_{R_{(y,x)}}$$

and

$$E(y) = E_{T_{(y,x)}} + E_{R_{(a,y)}}$$

The theorem holds since $E(x)$ is not increased; its smallest value is

$$E_{R_{(y,x)}} = E_R$$

which is a constant due to equation 7.2.

Inductive step: We assume that the theorem holds in tree T for $n = k$, where $k > 2$. We prove the theorem holds for $n = k + 1$. From the assumption, all the internal nodes in T have the least energy consumption in sending and relaying data since they all have the same property as node x in the previous procedures. At $n = k + 1$, since node x is an internal node, x has the least energy consumption in sending and relaying data in T. When adding node a to T, since

$$w(a,x) < w(a,y)$$

and

$$w(a,x) + P(a,x) > w(a,y) + P(a,y)$$

node a will be a child of node y instead of be the child of x in T'. Therefore, the energy consumption of node x is not increased in T'; x still has the least energy consumption in sending and relaying data in T'. Thus, the theorem holds. ■

Examples of applying the EHAEC are shown in Figs. 7.5(1-7). Figure 7.5(1) shows a unit weighted graph $G =< V, E >$, where $V = \{BS, a, b, c, d, e, f\}$, vertex BS is the root of the tree, and the other vertexes are sensor nodes that must be connected to the tree. Note that there are other edges in graph G, for example, (BS, f), (b, c), and (d, f). However, these edges have large weights, so we have ignored them. The weight is shown for each edge, for example, $w(a, b) = 240$ means that the weight of edge (a, b) is $240\mu J$. These values are calculated using Equations 7.1 and 7.2. We can now show how EHAEC forms a route. In the first step shown in Fig. 7.5(1), $u = r = BS$ because of Lines 5 and 7, hence the starting vertex of the tree forming procedure is BS. Therefore, BS is on the tree. Since $u = r = BS$ in line 11, lines 12 to 15 are not executed. For every vertex $v \in Adj[BS] = \{a, b, c\}$, line 16 judges whether $w(a, BS)$, $w(b, BS)$ and $w(c, BS)$ are less than $key[a]$, $key[b]$ and $key[c]$, respectively. Line 16 is true since the key fields are infinities, hence, $\pi[a] = BS$, $\pi[b] = BS$ and $\pi[c] = BS$ in line 17; $key[a] = w(a, BS) = 213$, $key[b] = w(b, BS) = 220$ and $key[c] = w(c, BS) = 325$ in line 18. In step 2 shown in Fig. 7.5(2), $u = a$ in line 7 since $key[a] = w(a, BS) = 213$ is minimum in step 1. Line 8 stores edge (a, BS) to tree A and line 9 decides the direction of edge (a, BS) is $dir(a, BS)$ since $\pi[a] = BS$ in step 1. Since $\pi[u] = r = BS$ in line 11, lines 12 to 15 are not executed. line 16 judges whether $w(b, a)$, $w(c, a)$ and $w(d, a)$ are less than $key[b]$, $key[c]$ and $key[d]$,

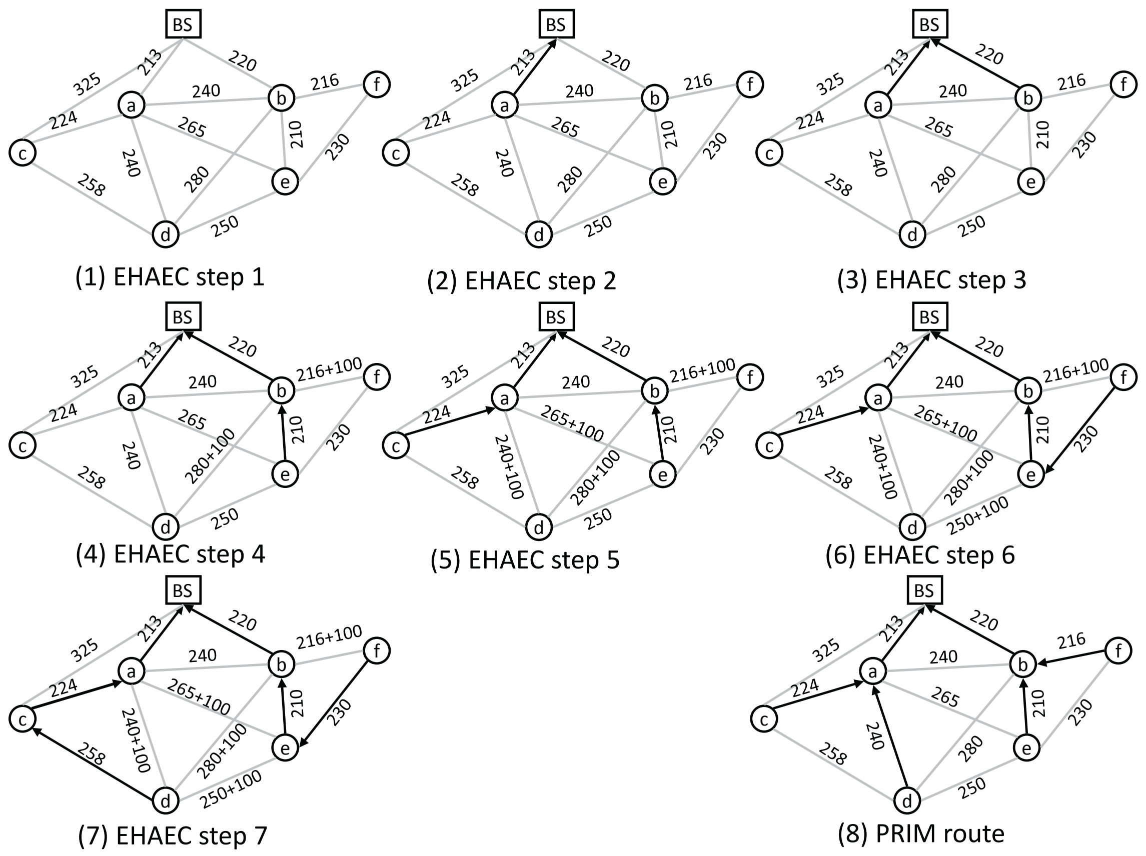


Figure 7.5: EHAEC and Prim Routing.

respectively. Line 16 is only true for vertexes c and d. Hence, $\pi[b] = BS$ which is not change, $\pi[c] = a$ and $\pi[d] = a$ in line 17; $key[b] = w(b,BS) = 213$ which is not change, $key[c] = w(c,a) = 224$ and $key[d] = w(d,a) = 240$ in line 18. The same procedure repeats until step 3 shown in Fig. 7.5(3). In step 4 shown in Fig. 7.5(4), after $u = e$ is extracted in line 7, $s \in Adj[\pi[u]] - u = Adj[\pi[e]] - e = Adj[b] - e = \{d, f\}$ in line 11. Therefore, $P(d,b) = E_R(e,b) = 100$ and $P(f,b) = E_R(e,b) = 100$ in line 12. $w(d,b)$ and $w(f,b)$ are updated in line 13. Line 14 judges whether $key[d]$ is $w(d,a)$, and $key[f]$ is $w(f,b)$, where $w(d,a)$ and $w(f,b)$ are values in the previous step. Since it is true that $key[f]$ is $w(f,b)$ which in the previous step, $key[f] = w(f,b)$ in line 15, the $w(f,b)$ here is the value calculated in line 13, hence the key field is updated. The rest part of step 4 is similar with that in step 1 and 2. The same procedure repeats in step 5, 6 and 7. In step 7 shown in Fig. 7.5(7), since all the nodes are connected by the EHAEC, a spanning tree has been formed. Compared with the route generated by Prim's algorithm in Fig. 7.5(8), EHAEC avoided the energy holes in nodes a and b.

The PRIM algorithm is optimal since it minimizes total energy consumption for data communications in a WSN by producing a minimum spanning tree route. In contrast, EHAEC is not optimal because it does not always minimize total energy consumption for the entire network. However, EHAEC does minimize energy consumption for all the internal nodes. In tree T constructed by EHAEC, except for the base station, let $I \subset T$ and $L \subset T$ be the sets of all internal nodes and all leaf nodes in T, respectively. Due to Theorem 1, since all the internal nodes consume the least energy needed for sending and relaying data, the total energy consumption of nodes in set I is minimum, hence, EHAEC is optimal in set I. Note that EHAEC is not optimal in set L since some leaf nodes may use phony weights to select a longer route for data communications in order to make EHAEC optimal in set I. This increases the energy consumptions of the leaf nodes. To compare EHAEC and PRIM we prove the following two theorems, one showing the condition of that EHAEC outperforms PRIM in avoiding energy holes and the other showing the energy loss in EHAEC.

Theorem 2 *For $G =< V, E >$, $\forall u, v, w \in V$. If $d_{(v,w)} < \sqrt{(n_u - n_w)\frac{E_{elec}}{\varepsilon_{amp}} + d^2_{(v,u)}}$, then edge (v, w) with $dir(v, w)$ will be selected by EHAEC to avoid the energy hole that might exist in node u if edge (v, u) with $dir(v, u)$ is selected by using the PRIM algorithm without phony weight. Here, d is the distance between two nodes and n_u and n_w are the number of child nodes of nodes u and w, respectively.*

Proof: Since

$$d_{(v,w)} < \sqrt{(n_u - n_w)\frac{E_{elec}}{\varepsilon_{amp}} + d^2_{(v,u)}}$$

holds, we have

$$d^2_{(v,w)} < (n_u - n_w)\frac{E_{elec}}{\varepsilon_{amp}} + d^2_{(v,u)}.$$

Because $E_{elec} * k$ and $\varepsilon_{amp} * k$ are constants,

$$E_{elec} * k + \varepsilon_{amp} * k * d^2_{(v,w)} < E_{elec} * k + \varepsilon_{amp} * k * d^2_{(v,u)} + (n_u - n_w) * E_{elec} * k$$

holds by multiplying $\varepsilon_{amp} * k$ and adding $E_{elec} * k$ to both sides of the inequation. Moreover, due to equation 7.1, we have

$$E_{elec} * k + \varepsilon_{amp} * k * d^2_{(v,w)} = E_{T_{(v,w)}}$$

and

$$E_{elec} * k + \varepsilon_{amp} * k * d^2_{(v,u)} = E_{T_{(v,u)}}$$

and due to equation 7.2, we have

$$E_{R_{(v,w)}} = E_{R_{(v,u)}} = E_R = E_{elec} * k$$

Therefore,

$$E_{elec} * k + \varepsilon_{amp} * k * d^2_{(v,w)} < E_{elec} * k + \varepsilon_{amp} * k * d^2_{(v,u)} + (n_u - n_w) * E_{elec} * k$$

becomes

$$E_{T_{(v,w)}} + E_{R_{(v,w)}} + n_w * E_R < E_{T_{(v,u)}} + E_{R_{(v,u)}} + n_u * E_R$$

Due to equation 7.3,

$$n_w * E_R = P(v,w)$$

and

$$n_u * E_R = P(v,u)$$

Therefore,

$$E_{T_{(v,w)}} + E_{R_{(v,w)}} + P(v,w) < E_{T_{(v,u)}} + E_{R_{(v,u)}} + P(v,u)$$

Moreover,

$$w(v,w) < w(v,u)$$

since

$$E_{T_{(v,w)}} + E_{R_{(v,w)}} + P(v,w) = w(v,w)$$

and

$$E_{T_{(v,u)}} + E_{R_{(v,u)}} + P(v,u) = w(v,u)$$

Therefore, due to EHAEC, edge (v,w) is selected to avoid the energy hole in node u, i.e., the theorem holds. ■

Theorem 3 *If routes R_P and R_E are created by PRIM and EHAEC, respectively, then R_E consumes more energy of $\varepsilon_{amp} * k * \sum(d^2_{(v,w)} - d^2_{(v,u)})$ than R_P, where edges (v,w) and (v,u) have the same definition as that in Theorem 2.*

Proof: Since regardless of the routing methods, the value of $\sum E_R$ in a route is a constant because the transmit data of every node can only be received once in a WSN. Compared with the PRIM algorithm an additional energy of $E_{T_{(v,w)}} - E_{T_{(v,u)}}$ is consumed by EHAEC in one changed edge. Evoking equation 7.1,

$$E_{T_{(v,w)}} - E_{T_{(v,u)}} = E_{elec} * k + \varepsilon_{amp} * k * d^2_{(v,w)} - E_{elec} * k - \varepsilon_{amp} * k * d^2_{(v,u)} = \varepsilon * k * (d^2_{(v,w)} - d^2_{(v,u)})$$

For all the edges changed in EHAEC, the total additional energy is

$$\varepsilon_{amp} * k * \sum(d^2_{(v,w)} - d^2_{(v,u)})$$

i.e., the theorem holds. ■

7.3 Route Switching Algorithms

In this section, we propose another two route switching algorithms called TINORESA and COMSA to solve the problem of finding switching routes.

7.3.1 Tired Node Resting Switching Algorithm

In this subsection we propose a tired node resting switching algorithm (TINORESA) to update the route obtained by EHAEC. TINORESA can find several routes to be alternatively applied with EHAEC until the lifetimes of nodes are balanced. In TINORESA, the concept of "tired nodes" is defined as the nodes that had larger than average energy consumption in the previous route. The "tired nodes" of the previous route are not allowed to relay data from child nodes in the current route in order to rest themselves. Accordingly, TINORESA applies EHAEC to form a spanning tree where the "tired nodes" do not consume spare energy. The input of TINORESA is the graph obtained by the previous route, and the output returns the new spanning tree A where the "tired nodes" are rested. Note that the first input is the route created by EHAEC. Algorithm 2 shows the procedure of TINORESA.

```
Algorithm 2: TINORESA(G,r)
01  Q ← PREV(G, r), where first PREV(G, r)=EHAEC(G,r)
02      for each $u \in Q$
03          $E(u) = E_{T_u} + nE_R$, where n is the number of child of u
04      $E_{ave} = \sum_{i=1}^{N} E(i)/N$
05      for each $v \in Q$
06          if $E(v) > E_{ave}$
07              $tired[v] \leftarrow v$
08              if $\pi[v]! = 0$
09                  $w(v,c) = \infty$, where c is the child candidate of v
10      EHAEC(G,r)
```

We show how TINORESA works. In line 01, all the vertexes that have the network connection information obtained by the previous route PREV(G, r) are stored in a priority queue Q. Lines 02~04 calculate the average energy consumption of nodes in EHAEC. Lines 05~09 show that, if the energy consumption $E(v)$ of a node v is bigger than the average energy consumption E_{ave}, the node u is defined as a "tired node," and the weight between node v and other nodes c, which are the child candidates of v, is set to infinity

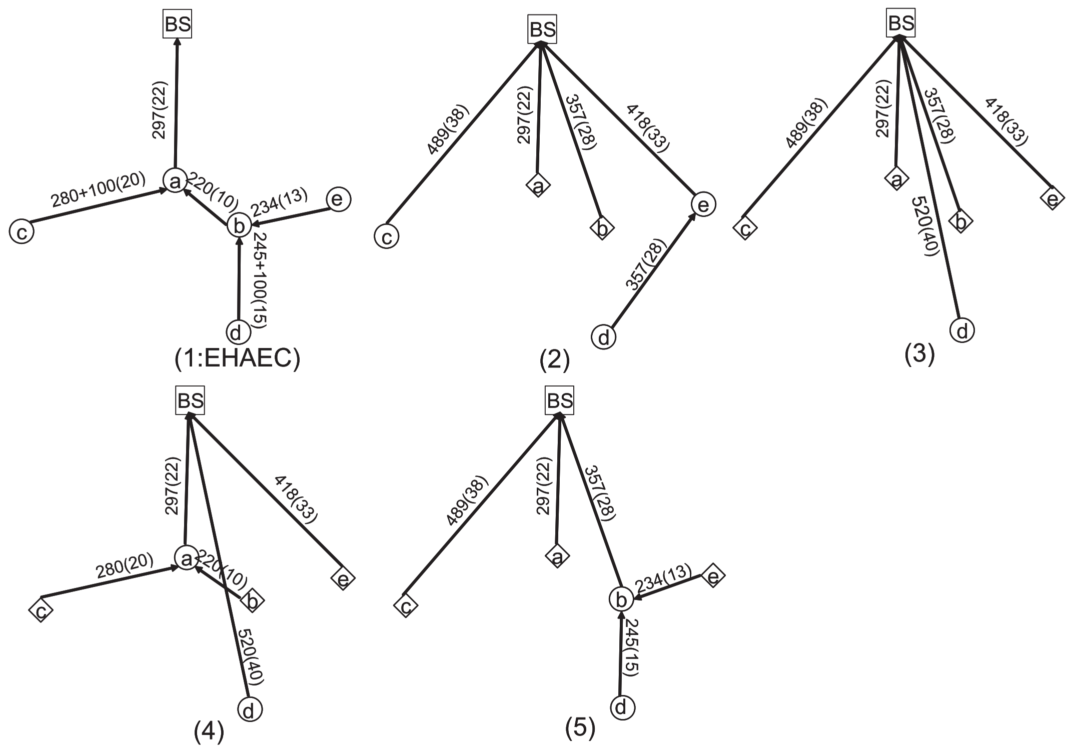


Figure 7.6: Route switching by TINORESA: rhombic nodes are tired nodes.

if node v is selected for the tree. Finally, using the updated information where “tired nodes” are rested, TINORESA calls EHAEC to create a new route. Since TINORESA goes through all the vertexes twice and applies EHAEC, the complexity of TINORESA is $O(2|V| + |E|\lg|V|)$. Moreover, since $2|V|$ is much smaller than $|E|\lg|V|$, the complexity is $O(|E|\lg|V|)$.

Figure 7.6 shows an example of TINORESA. We assume that there are five nodes and a base station in a network. In Fig. 7.6(1), after several rounds of communication, nodes a and b are found to be “tired nodes”. Therefore, TINORESA forms a new route in Fig. 7.6(2) where nodes a and b are not the parents for any nodes and thus do not relay the data, i.e., energy is saved. Similarly, after another several rounds of communication, other nodes are found to be “tired”, and thus the route in Fig. 7.6(3$\sim$5) is formed by TINORESA to avoid energy holes until all nodes run out of energy.

However, a problem came to light: how often should TINORESA switch the route? A trade-off should be considered in answering this question. Generally, more frequent route switching can reduce the difference in the remaining energy between the “tired nodes” and the other nodes; however, the route switching overhead is high. In contrast, less frequent route switching may increase the difference in remaining energy but the overhead is low. As the timing of the route switching, we use a threshold of the average remaining energy capacity such as route switching occurs at the time of 80%, 50% or 30% of the average energy capacity remains. Moreover, each node periodically sends information with the sensing data informing the base station how much energy remains in the node.

7.3.2 Complementary Switching Algorithm

In this subsection we propose a complementary switching algorithm (COMSA), which attempts to find only one switching route that can be alternately used with the route created by EHAEC, where the energy consumption of each node in COMSA is the complementary value of that in EHAEC. We define a complementary spanning tree T_C as follows. In the graph $G = < V, E >$, $E(v)$ denotes the energy consumed by node $v \in V$. $T_E = < V, E_E >$ is a spanning tree for the G obtained by EHAEC. The complementary tree $T_C = < V, E_C >$ is defined as a spanning tree for G, where $E(v) + E'(v) = C(v)$. We call $C(v)$ the complete value and $E'(v)$ the complementary value of $E(v)$. Moreover, $max\ C(v) - min\ C(v) \leq \Delta$ for a constant Δ. If $\Delta = 0$, we call T_C a complete complementary spanning tree. We present COMSA to build T_C from T_E as follows.

```
Algorithm 3: COMSA(G,r)
01  Q ← V(G)
02  $C = E(EHAEC_{MAX}) + E_{min}(EHAEC_{MAX})$
03  $\Delta$ ← TradeoffValue
04  **for** each $u \in Q$
05      $E(u) \leftarrow \infty$
06      $\pi[u] \leftarrow NIL$
07  $E(r) \leftarrow 0$
08  **while** $Q \neq 0$
09      u ← EXTRACT-MIN-E(Q)
10      **for** each $v \in Adj[u]$
11          **if** $\exists$ $v$, that if $\pi[v] = u$ then $E(u) \in (C - E_{EHAEC}(u) \pm \Delta)$
12              $\pi[v] \leftarrow u$
13              $T_C \leftarrow \{(v, \pi[v]) : v \in V - r - Q\}$
14              edge direction: dir(v,u)
15  **return** $T_C$
```

Next we explain how COMSA works in accordance with Algorithm 3. Lines 1-7 show the initializations, which differ from those in EHAEC in lines 2, 3, 5, and 7. Line 2 defines the complete value C, which is the sum value of the energy consumed by the most energy consuming node in EHAEC and the achievable minimum energy consumption for the same node. Line 3 defines a trade-off value for C, since it is hard to ensure C is always the same due to the different power consumption of each node. Lines 5 and 7 initialize the energy consumption of vertex u and r, where r is the base station (which is the root of the tree). Lines 8-15 show the procedure for finding the switching route. Line 9 extracts

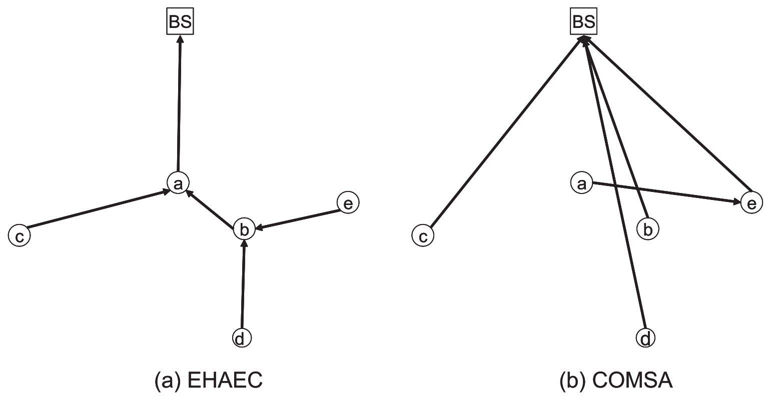


Figure 7.7: Routes of EHAEC and COMSA.

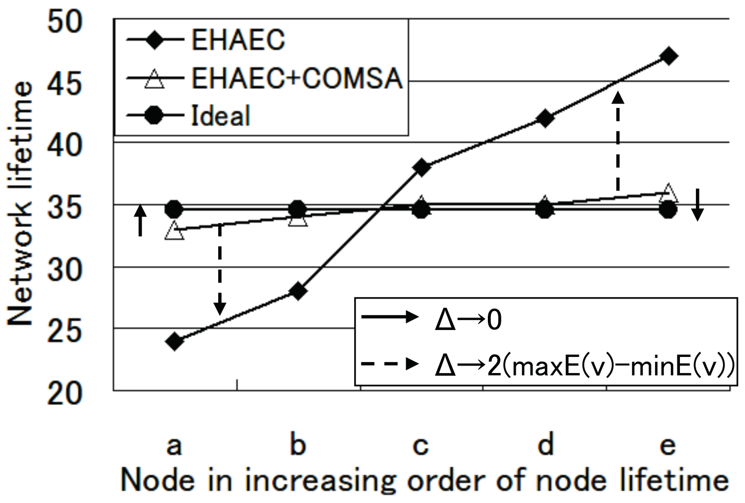


Figure 7.8: The use of theoretical bound of Δ of example in Fig.7.7.

a vertex $u \in Q$ where the energy consumption of u is the smallest. Lines 10-14 show that for every vertex v adjacent to u but not in the tree, if v is the child of u, the energy consumption of u should be the complementary value of that in EHAEC; thus the π field is updated as u is the parent of each vertex v. Line 13 updates the tree T_C and line 14 defines the direction of the edges. Finally, after the iteration, COMSA returns the tree T_C. The complexity of COMSA is the same as that of EHAEC, i.e., $O(|E|\lg|V|)$.

Figure 7.7 shows the routes made by EHAEC and COMSA for a 6-node WSN, where the base station is the root of a tree. The two routes are applied in the same communication rounds alternately to obtain uniform energy consumption. However, a problem is that if the value of Δ is too large, the meaning of complementary value does not exist anymore since it would not balance the energy consumption of nodes by utilizing EHAEC and COMSA alternatively. We next prove the following theorem for a bound of Δ.

Theorem 4 *In COMSA, Δ is bounded as $0 \leq \Delta < 2(max\, E(v) - min\, E(v))$ to maintain the meaning of complementary value. Where $v \in V$, $max\ E(v)$ and $min\ E(v)$ are the energy consumption of the most and least energy consuming node in EHAEC, respectively.*

Proof: Since

$$0 \leq \Delta < 2(max\, E(v) - min\, E(v))$$

we have

$$0 \leq \frac{\Delta}{2} < max\, E(v) - min\, E(v)$$

Due to the explanation of complementary value in the first paragraph of this subsection, if we let

$$max\ C(v) - min\ C(v) = \Delta$$

the inequation becomes

$$0 \leq \frac{max\ C(v) - min\ C(v)}{2} < max\ E(v) - min\ E(v)$$

Since

$$C(v) = E(v) + E'(v)$$

holds, where $E(v)$ and $E'(v)$ are the energy consumptions of node v in EHAEC and COMSA, respectively, $\frac{max\ C(v) - min\ C(v)}{2}$ means the energy consumption difference between the most and least energy consuming node where EHAEC and COMSA are utilized alternatively (we express this condition by using the notation EHAEC+COMSA). Moreover, $max\ E(v) - min\ E(v)$ means the energy consumption difference between the most and least energy consuming node in EHAEC.

The meaning of complementary value is maintained if and only if the energy consumption difference of EHAEC+COMSA is less than that of EHAEC. This is because the energy consumption of nodes becomes more uniform as the energy consumption difference becomes smaller. Since

$$0 \leq \frac{max\ C(v) - min\ C(v)}{2} < max\ E(v) - min\ E(v)$$

holds, the meaning of complementary value is maintained; thus, the theorem holds. ■

We show how the theoretical bound of Δ affects the network lifetime. Figure 7.8 shows the network lifetimes of the WSNs in Fig. 7.7 by using EHAEC, EHAEC+COMSA, and the estimated ideal network lifetime ("Ideal" in Fig. 7.8) when there are no energy holes. Generally, the network lifetime of EHAEC+COMSA is between those of EHAEC and Ideal. As Δ approaches 0, the network lifetime of EHAEC+COMSA approaches that of the Ideal since the energy consumptions of nodes in COMSA are closer to the complementary energy consumptions of nodes in EHAEC. In contrast, as Δ approaches $2(max\, E(v) - min\, E(v))$, the network lifetime of EHAEC+COMSA approaches that of EHAEC since the energy consumptions of nodes in COMSA are closer to those in EHAEC. This is because COMSA balances the remaining energy of the nodes.

7.3.3 Route Switching Overheads

In this subsection, we will discuss the route switching overhead in the TINORESA and COMSA algorithms. Generally, two kinds of overheads exist in route switching algorithms: cost incurred by sending route switching information and cost incurred by updating the routing table.

More specifically, the first type of overhead is the energy consumed by sending route switching information from the base station to all the nodes in the network. The base station should generate reverse data communication based on the previous route to tell every node their new parent node's address information before the route switching since the base station is the only node that knows the global address information. Generally, the overhead is less than the energy consumed in the previous data communication since the energy consumed at base station is not included.

The second type of overhead is the energy consumed by switching the communication route for each node, i.e., the energy consumption of updating the routing table. The routing table will be updated if the data transmitting relation between nodes has been changed by route switching. The energy consumed by changing nodes' connections is difficult to calculate since it depends on the energy consumed by program instructions. Therefore, we assume that it consumes energy of x for one connection change, so the overhead can be calculated by using x. Overall, TINORESA has higher overheads than COMSA since it performs more route switching operations.

7.3.4 Node leaving and joining the network

Next we describe what happens for the proposed algorithms when a node leaves or joins the network. First, if a node leaves the network, the base station can no longer receive its data or that of its child nodes. The base station thus considers the node to be terminated and rebuilds the route to eliminate the terminated node by using one of the proposed algorithms. The energy consumption of each node changes due to changes in the route. However, route rebuilding is necessary only for an internal node termination since the termination of a leaf node does not affect the data communications of the other nodes in the network. Second, if a node joins the network, the base station adds it to the data communication route. The procedure for doing this comprises three steps. (1) The new

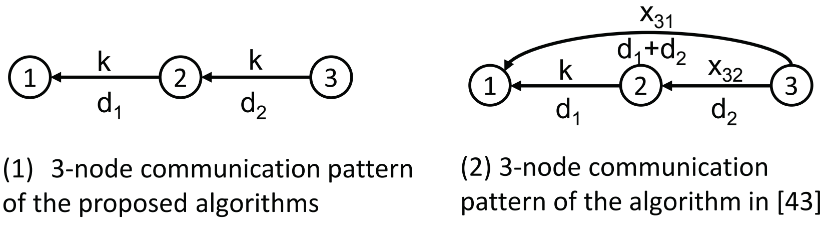

k
k
d1
d2
(1) 3-node communication pattern of the proposed algorithms
x31
d1+d2
k
x32
d1
d2
(2) 3-node communication pattern of the algorithm in [43]

Figure 7.9: Communication patterns of the proposed algorithms and the algorithm in [37].

node sends a trigger message to the base station informing the network of its entry into the network. (2) Upon receiving the message, the base station terminates usage of the current data communication route in the network and re-executes one of the proposed algorithms to generate a new route that includes the new node. (A more efficient approach would be to rebuild only part of the route instead of generating a new route. This remains for future work due to its complexity.) (3) The network begins using the new route for data communications.

7.3.5 Comparison with previous work

We compare the proposed algorithms with another algorithm proposed in [37] since it can solve the energy hole problem in an energy efficient manner and it uses the same energy consumption model used in this thesis. The authors of [37] assumed that all sensor nodes are uniformly distributed in a plane and that nodes are located at the center of imaginary hexagons, so the distances between neighbor nodes are the same. Their proposed algorithm enables every node in that network topology to send its data to intermediate nodes by dividing the data into fractions so that the node energy consumptions are fairly evenly distributed. Figure 7.9 shows the communication patterns of our algorithms and the algorithm in [37] for a three-node network. The characters above and beneath the lines represent the amount of data transmitted and the transmission distance, respectively. Figure 7.9(2) shows that node 3 transmits its data by transmitting x_{31} and x_{32} separately using two routes, $3 \rightarrow 2 \rightarrow 1$ and $3 \rightarrow 1$. Since we and the authors of [37] assume that every node generates the same amount of data to send, $k = x_{32} + x_{31}$ in Fig. 7.9. Moreover, in spite of data aggregation energy consumption in [18] to match the assumption of this thesis, the energy consumption difference of data communications which use the algorithms proposed in [37] and this thesis is the energy consumption that depends on the communication distance, which is represented by $\varepsilon_{amp} * k * d^2$ in equation 7.1. Comparison of $\varepsilon_{amp} * k * d^2$ in Fig. 7.9(2) with that in Fig. 7.9(1) shows that the route over-consumes $\varepsilon_{amp} * x_{31} * [(d_1 + d_2)^2]$ for transmitting x_{31} over distance $d_1 + d_2$ and under-consumes $\varepsilon_{amp} * x_{31} * d_2^2$ for transmitting x_{31} over distance d_2. Therefore, an extra amount of energy, $\varepsilon_{amp} * x_{31} * [(d_1 + d_2)^2 - d_2^2]$, is consumed by the route shown in Fig. 7.9(2). Generally, for an N-node network, the algorithm in [37] consumes extra energy $E_{extra} = \sum^N \varepsilon_{amp} * x_{non-one-tran} * [d^2_{non-one-tran} - d^2_{one-tran}]$ compared with the algorithms proposed in this thesis. The $x_{non-one-tran}$ represents the sizes of the data fractions sent to nodes other than the nearest neighbor node, for example x_{31} in Fig. 7.9(2), the $d_{non-one-tran}$ is the distance between the two nodes between which $x_{non-one-tran}$ is transmitted, and the $d_{one-tran}$ is the distance between the nearest neighbor nodes between which the data fractions are transmitted, for example x_{32} in Fig. 7.9(2). Since $d_{non-one-tran}$ is always greater than $d_{one-tran}$, E_{extra} is always positive, our proposed algorithms have better performance than the algorithm in [37] regardless of whether the distances between neighbor nodes are the same or not.

Chapter 8

Fault Tolerant Algorithms

In this chapter, we present two kinds of proactive fault tolerant algorithms to tolerate node failures in WSNs. The first algorithm attempts to maintain k-connectivity of a network, whereas the second designates backup nodes for only critical nodes which are easy to fail in the network.

8.1 EHAEC for One-Fault Tolerance

Our proposed algorithm EHAEC for one-fault tolerance (EHAEC-1FT) maintains two-connectivity of the network, because it can tolerate the failure of one node in the WSN. When the requirement is more than two-connectivity, it can be extended to X-nFT, which can be used by any spanning tree routing algorithm X for n-fault tolerance.

```
Algorithm 4: EHAEC-1FT(A,w,r)
01 for each $u \in A$
02     $key[u] \leftarrow \infty$
03 for each $y \in A \backslash \{x\}$, where $x \in A$ and $\pi[y] = x$
04     for each $z \in Adj[x] - y$
05         for each $s \in Adj[\pi[y]] - y$ and $s \in A \backslash 1FTSG, s \neq r, y \neq r, \pi[y] \neq r$
06             $P(s, \pi[y]) = \sum_{a \in 1FTT} E_R(a, \pi[a])$, where $\pi[a] = \pi[y]$
07             $w(s, \pi[y]) = w(s, \pi[y]) + P(s, \pi[y])$
08             if $s \in Q$ and $\pi[s] = \pi[y]$ and $key[s]$ is $w(s, \pi[y])$ that in last round
09                 $key[s] \leftarrow w(s, \pi[y])$
10         if $z \in A \backslash \{x, y\}$ and $w(z, x) < key[z]$
11             $\pi[z] \leftarrow x$
12             $key[z] \leftarrow w(z, x)$
13     $1FTSG \leftarrow \{(z, \pi[z]) : z \in A \backslash \{r, x, y\}\}$
14     edge direction: $dir(z, \pi[z])$
15 return $1FTT = A \cup 1FTSG$
```

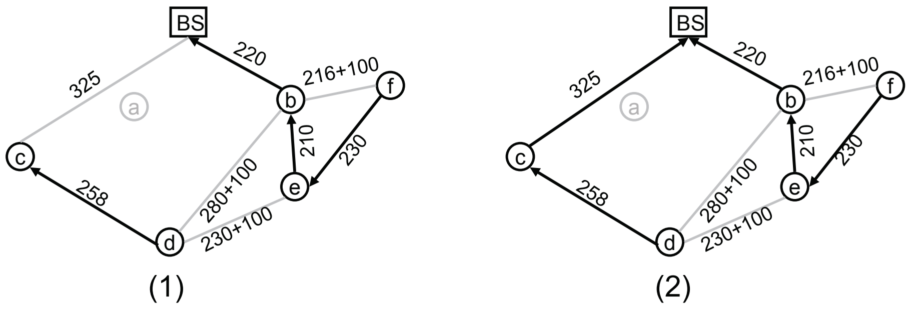


Figure 8.1: EHAEC-1FT example.

Table 8.1: 1FTSG lookup table

Failed node	Affected nodes	Added edge
a	c,d	(c, BS)
b	e,f	(e, d)
c	d	(d, a)
d	NIL	NIL
e	f	(f, b)
f	NIL	NIL

EHAEC-1FT uses EHAEC to find a subgraph of G that, in union with the graph of set A obtained by EHAEC, can maintain two-connectivity and one-fault tolerance. We call this subgraph a "one-fault tolerant subgraph (1FTSG)" and the union graph a "one-fault tolerant tree (1FTT)". The input to EHAEC-1FT is graph A of EHAEC, and the output is the union graph 1FTT. The basic idea of EHAEC-1FT is to first suppose that every node in A has failed, and then find new parent nodes for the child nodes that belonged to the failed nodes. Note that if a node fails, all the edges were connected to the failed node are deleted from the tree. In EHAEC-1FT, Lines 1 and 2 initiate the edge weights. Line 3 identifies the child nodes of the failed nodes, and Lines 4-14 use EHAEC to find new parent nodes for only these child nodes. Note that Lines 4-12 and 13-14 in EHAEC-1FT have the same function as Lines 10-18 and 8-9 in EHAEC, only the parameters are different. Finally, Line 15 returns tree 1FTT. Because Lines 3-4 are executed $O(|V|)$ times, and Lines 5-15 are executed $O(|E| \lg |V|)$ times, the complexity of EHAEC-1FT is $O(|V||E| \lg |V|)$.

The EHAEC-1FT example in Fig. 8.1 uses the output A of EHAEC that is shown in Fig. 7.5(7). Figure 8.1(1) shows the graph when node a has failed. The edges connected to a are deleted because of this failure. The failure of node a means that c and d are removed from the communication tree. Therefore, node c takes BS as its new parent node and maintains the connectivity of the communication tree (Lines 4-14). Therefore, edge (c, BS) with $dir(c, BS)$ is added to subgraph 1-FTSG in Fig. 8.1(2). Similarly, if nodes

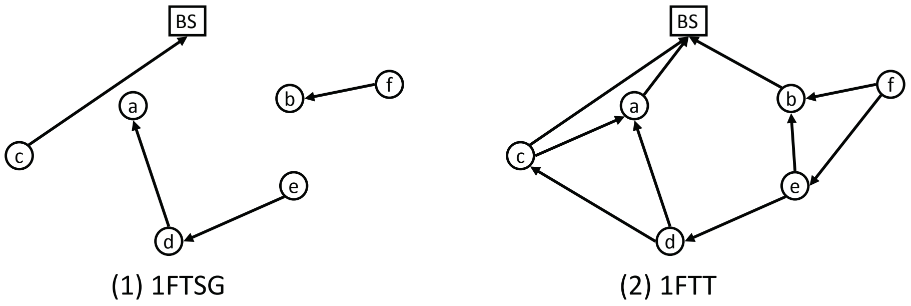


Figure 8.2: Graph of 1FTSG and 1FTT.

$b, c, d, e, and\ f$ failed in turn, new edges would be added to 1-FTSG, as shown in Table 8.1. Consequently, 1FTSG is constructed as in Fig. 8.2(1). Moreover, the combination of 1FTSG with the tree formed by EHAEC constructs a 1FTT (Line 15), as shown in Fig. 8.2(2).

```
Algorithm 5: X-nFT(A,r)
01  for $i = 1$ to $n$
02      for all $y_i \in A \backslash \{x_i\}$, where $x_i \in A$ and $\pi[y_i] = x_i$
03          do TreeRoutingAlgorithm(A,r)
04              return $iFTSG$
05  return $nFTT = A \cup 1FTSG \cup 2FTSG \cup \ldots \cup nFTSG$
```

As mentioned above, EHAEC-1FT can be extended to X-nFT to maintain n-fault tolerance for any tree routing algorithm X. EHAEC-1FT can be extended to satisfy the requirement of tolerating an n-node failure, as shown in Algorithm 5. X-nFT finds all the child nodes of n failed nodes (Lines 1 and 2), and then uses a particular tree routing algorithm (Line 3) to build new connections between these child nodes and other nodes in the tree, resulting in n-fault tolerance. For every $i = 1...n$, X-nFT returns an i-fault tolerant subgraph, $iFTSG$ (Line 4). Finally, X-nFT returns the union graph $nFTT$, which is n-fault tolerant.

Different systems have different fault tolerant redundancy requirements. There are two ways to make a system fault tolerant: create a fault tolerant redundancy when or before a failure (standby fault tolerance, STB), or constantly have a fault tolerant redundancy (active fault tolerance, ACT). With STB, the WSN uses tree A of EHAEC for data communication. If n nodes fail, only the parts of subgraphs 1FTSG...nFTSG that can tolerate the failed nodes are used with tree A. ACT uses tree nFTT, which is a complete combination of tree A and the fault-tolerant subgraphs.

8.2 Active Spare Selecting Algorithm

We considered another provisioned tolerance scheme that finds spare nodes for the failed node, which serve as cut vertexes. In this situation, the spare node takes the responsibility of the failed node. In most related works, spare nodes are additional nodes that placed at right positions accompany with the failed nodes. However, in WSN systems for infrastructures such as bridges and tunnels, it is difficult to place additional nodes because we do not know which node is likely to fail. Placing spares around some critical node is risky. Moreover, additional spare nodes increase costs. Therefore, in this section, we present the active spare selection algorithm (ASSA), which attempts to select appropriate nodes in the network to be used as active spare nodes. The spare nodes only substitute the failed critical nodes. Here, critical nodes are nodes that act as cut-vertexes in the network, i.e., the internal nodes. When a critical node fails, the ASSA selects an active spare node to take on its responsibilities.

```
Algorithm 6: ASSA(A,$\alpha$,$\beta$)
01 input impact factor $\alpha$ and $\beta$, where $\alpha + \beta = 1$
02 for each $u \in A \backslash L$
03     $CL(u) = NumOfSuccessor(u)$
04 for each $y \in A \backslash L$, where $\pi[y] = z$
05     for each $x \in Adj[y]$
06         $CL\_TEMP[x] = CL(x) + CL(y) + 1$
07         $w(x,y) = E_{T_{(x,y)}} + E_{R_{(x,y)}}$
08     calculate $NCL = \frac{1}{RMS[CL\_TEMP]}$ and $Nw = \frac{1}{RMS[w]}$
09     for each $x \in Adj[y]$
10         $NEC(x,y) = \alpha * NCL(x) + \beta * Nw(x,y)$
11 for each $z \in A \backslash L$, where $\pi[y] = z$, and $\pi[z] \neq x$
12     if $z$ failed
13         for each $y \in A \backslash L$, where $\pi[y] = z$, start the loop with $CL(y)$
           in decreasing order
14             if $\exists x \in A \backslash L$, where $NEC(x,y)$ is minimum
15                 $\pi[y] = x$
16                 $CL(x) = CL\_TEMP(x)$
17                 $PRCT \leftarrow \{(y, \pi[y])\}$
18 return $PRCT$
```

In the ASSA, we use the concept of a sensor node's critical level $CL(x)$, which is the number of successor nodes to x ($CL(x) = NumOfSuccessor(x)$). Generally, a node is more critical if it transmits data from more child or successor nodes. Therefore, it has a higher critical level if it has more successors. Moreover, because the ASSA is designed to select active spares using a combined weight of the node's critical level and edge weight, we also use an evaluation criterion $EC(x,y) = \alpha * CL(x) + \beta * w(x,y)$. Suppose that node z fails and y is a child node of z. For a spare node candidate x, the evaluate criterion considers the weight of the critical level of node x and the edge weight $w(x,y)$. Because $CL(x)$ and $w(x,y)$ are different metrics, we normalize the values using $N = \frac{1}{RMS[x]}$ so that they are comparable. $RMS[x] = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i)^2}$ indicates the root mean square, which is a common normalization method. Hence, $CL(x)$ and $w(x,y)$ are normalized to $NCL(x)$ and $Nw(x,y)$, respectively, and the normalized $EC(x,y)$ is $NEC(x,y) = \alpha * NCL(x) + \beta * Nw(x,y)$. Moreover, we set $\alpha + \beta = 1$. System designers can set the values of α and β to express their inclination towards fault tolerance or energy efficiency.

We can now explain how the ASSA works. First, in Line 1, the values of α and β are chosen by the system designer. Line 3 initializes the critical levels of all nodes u in tree A (except the leaf node set L). Lines 4 to 7 define a temporary critical level value ($CL\_TEMP[x]$) for each node x that is a neighbor to node y such that y's current parent node is z. We set $CL\_TEMP[x]$ in this way so that, if node y's parent node z fails and node y takes node x as its new parent, then the critical level of x is updated. Line 7 sets the edge weight $w(x,y)$, which indicates the energy consumption for data communication between nodes x and y. Note that $w(x,y)$ does not include the phony weight, because the ASSA uses the critical level. Line 8 calculates the normalized CL_TEMP and $w(x,y)$. Lines 9 and 10 calculate the normalized evaluation criterion, $NEC(x,y)$, which determines if x should be selected as be the active spare node of z. In Lines 11 to 17, if any node z has failed, all its child nodes (y) will find the new proper parent node x that has the minimum $NEC(x,y)$. This procedure starts with node y in descending order of $CL(y)$. Line 15 sets x as the new parent node of y, and Line 16 updates $CL(x)$. Line 17 stores the new link relation of $\{(y, \pi[y])\}$ to $PRCT$, where $PRCT$ is defined as the partition route changed tree. Finally, Line 18 returns $PRCT$. We calculated the complexity of ASSA in three parts. First, Lines 2 and 3 are executed $O(|A-L|)$ times. Next, for Lines 4-10, Line 4 is executed $O(|A-L|)$ times, and Lines 5-7 and 9-10 are executed $O(|E|)$ times. Therefore, the second part is executed $O(|E||A-L|)$ times. In the third part, Line 11 is executed $O(|A-L|)$ times, and Lines 13-17 are executed $O(|A-L|)$ times. Therefore, the third part is executed $O(|A-L|^2)$ times. Overall, the complexity of ASSA is $O(|A-L|^2)$.

We now give an example that uses ASSA, as in shown in Fig. 8.3. Figure 8.3(1) shows a tree route generated by a spanning tree algorithm. In Fig. 8.3(2), an internal node b has failed. Because active spares should be selected so that node c can tolerate the failure of node b, the normalized evaluate criterion $NEC(x,y)$ for each possible link between the child nodes and their neighbors are calculated in Line 10. By comparing the values of $NEC(x,y)$ for different α and β, the appropriate spare nodes are found using lines 14-17. Figures 8.3(3-5) show the new routes built by adapting active spares when

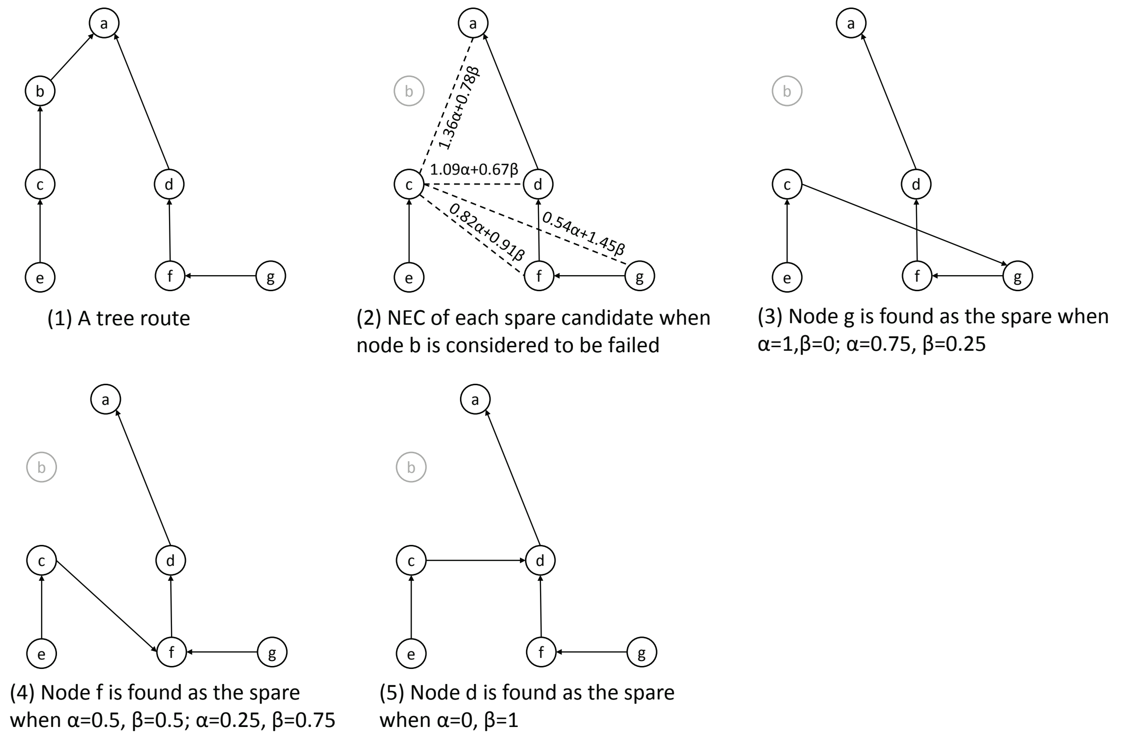

1.36α+0.78β
1.09α+0.67β
0.82α+0.91β
0.54α+1.45β
(1) A tree route
(2) NEC of each spare candidate when node b is considered to be failed
(3) Node g is found as the spare when α=1,β=0; α=0.75, β=0.25
(4) Node f is found as the spare when α=0.5, β=0.5; α=0.25, β=0.75
(5) Node d is found as the spare when α=0, β=1

Figure 8.3: ASSA example.

$\alpha = 1$ and $\beta = 0$, $\alpha = 0.75$ and $\beta = 0.25$, $\alpha = 0.5$ and $\beta = 0.5$, $\alpha = 0.25$ and $\beta = 0.75$, and $\alpha = 0$ and $\beta = 1$, respectively.

The ASSA considers both fault tolerance and energy efficiency, and can be adapted to different applications. First, we must consider the situations when either fault tolerance or energy efficiency is more important. Energy efficiency is considered more important when the WSN environment is stable. For instance, when there are no or very few signal interferences, sensor nodes are rarely damaged, or there are short communication distances. Fault tolerance is considered more important in the opposite case, when the environment is not stable, or when sensor nodes are not independent of each other. For instance, if an internal node only can send its data when it has already received its child nodes' data. Secondly, we also considered some concrete situations of ASSA, for example, when the number of nodes is large, or a high density of nodes in a WSN. If the number of nodes is large, it is easier to find child nodes, and $NCL(x)$ is relatively large. On the contrary, the communication distance is short if the number of nodes is large or the WSN is dense, so $Nw(x,y)$ is small. Therefore, $NCL(x)$ is the dominant factor that affects the value of $NEC(x,y)$ if $\alpha \neq 0$, in this case. Hence, if the energy efficiency is more important than fault tolerance, $\alpha = 0$ and $\beta = 1$ are effective. Alternatively, we may have a low density or small WSN. In this case, $NCL(x)$ is small and $Nw(x,y)$ is large. Consequently, we can set $\alpha = 1$ and $\beta = 0$ if a WSN requires fault tolerance more than energy efficiency.

Chapter 9

Simulations

In this chapter, we evaluate the effectiveness of the proposed algorithms by simulating the lifetime of an entire WSN, and compare it with the lifetime of each node when using other algorithms. The input of the simulations is the graph $G =< V, E >$ of the network. In the simulations, we assumed each node has $0.5J$ of energy capacity. We calculated the lifetime of sensor nodes by first calculating the energy consumption of each node in different routes by using equations (1) and (2), and then calculate a node's data communication rounds (lifetime) as: $(node\ energy\ capacity)/(node\ energy\ consumption)$.

9.1 Simulations of Energy Hole Aware Energy Efficient Routing Algorithms

In this section, we compare the lifetime of each node in routes generated by the direct data transmission, the PRIM algorithm and our proposed EHAEC, TINIRESA and COMSA. In TINORESA and COMSA, we ignored the route switching overhead since the overhead value is not accurate because the second type overhead is hard to calculate. We show two simulation results for a 20-node WSN and a 50-node WSN. In the WSNs, the sensor nodes are randomly distributed such as Fig.9.1 shows.

Figure 9.2 shows the simulation results for a 20-node WSN and a 50-node WSN obtained by routes generated by the direct data transmission, the PRIM algorithm, EHAEC, TINIRESA and COMSA. Table 9.1 and table 9.2 show the algorithms' efficiency by the direct data transmission when 1%, 50% and 100% of nodes terminated in the two simulations. The simulation results show the same tendency as that in Fig. 7.2. Figure 9.2 shows that except for the last few nodes, PRIM, EHAEC, TINORESA, and COMSA perform much better than direct transmission in extending the communication rounds. EHAEC is more efficient than the PRIM algorithm in balancing the lifetime of each node. The re-

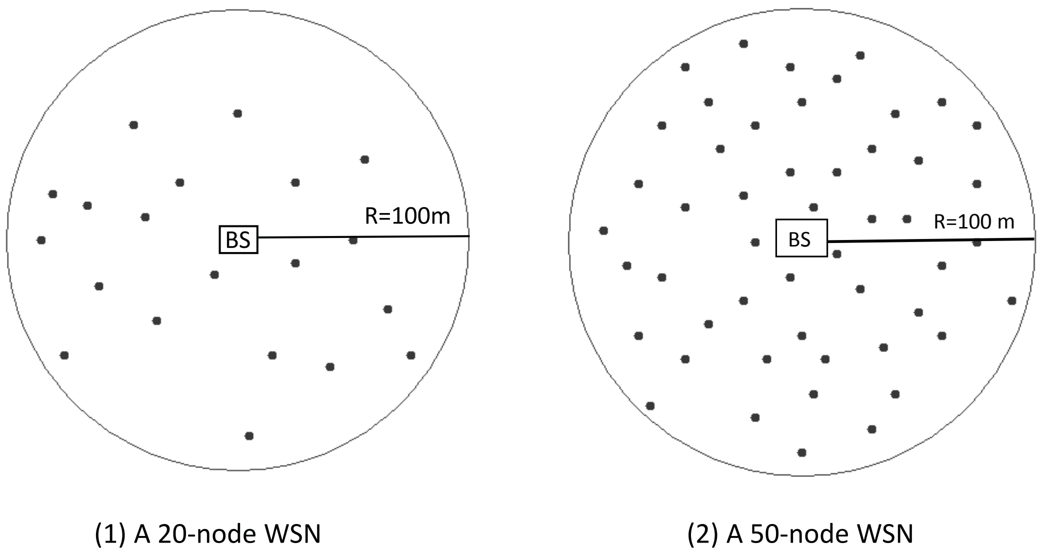


Figure 9.1: A 20-node and a 50-node WSN.

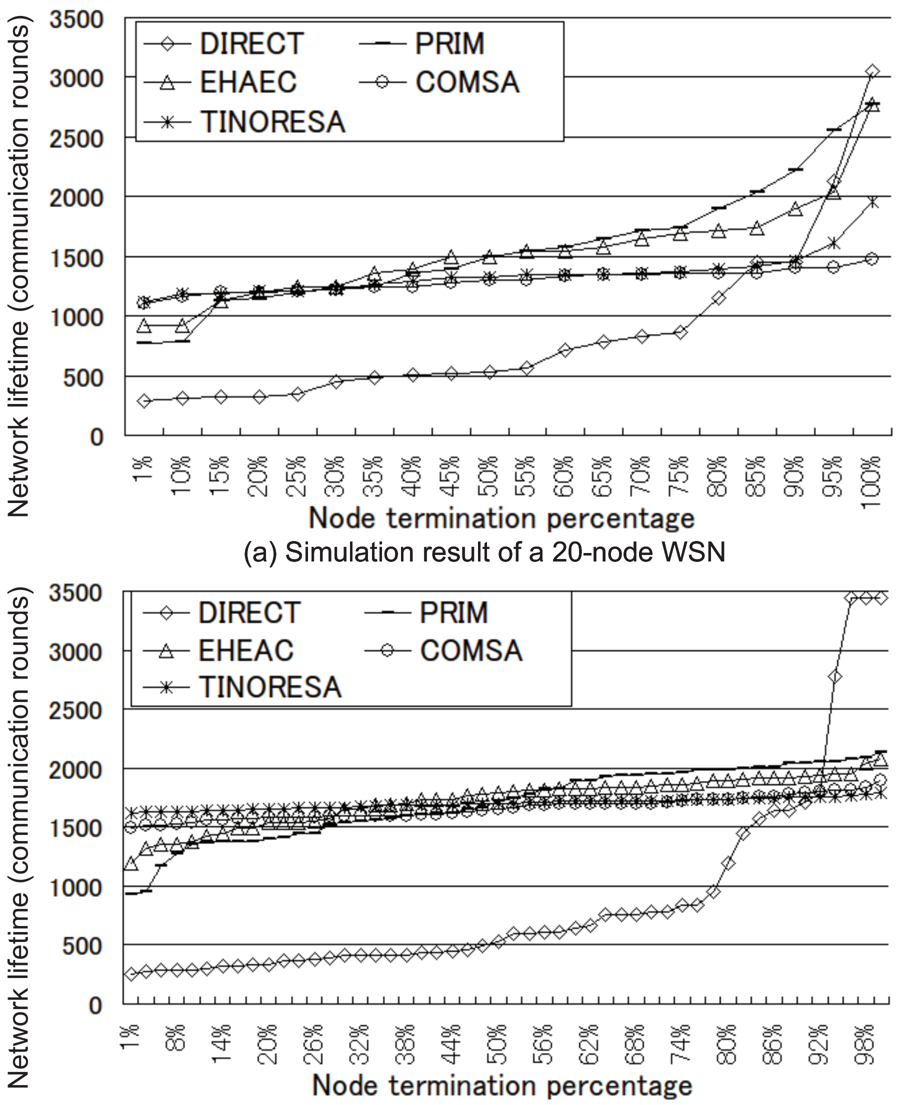
Table 9.1: Algorithms' efficiency by DIRECT in the 20-node network

	DIRECT	PRIM	EHAEC	TINORESA	COMSA
1%	1	2.7	3.2	3.8	3.8
50%	1	2.8	2.8	2.5	2.5
100%	1	0.9	0.9	0.6	0.5

sults also show that TINORESA and COMSA can balance the lifetime of each node, and that TINORESA is more efficient than COMSA. Notice that TINORESA could be less efficient than COMSA if the route switching overhead is considered into the simulation since TINORESA has a larger overhead than COMSA. From the standpoint of avoiding energy holes, our proposed EHAEC, TINORESA, and COMSA can extend the lifetime to about 3.2, 3.8, and 3.8 times, respectively, that for direct data transmission when 1% of the nodes are terminated in the 20-node network. And they can extend the lifetime to about 4.8, 6.5, and 6 times, respectively, that for direct data transmission when 1% of the nodes are terminated in the 50-node network.

Comparing the lifetime of the 50-node WSN with the 20-node WSN, we can see that our algorithms are more efficient in extending WSNs' lifetime and avoiding energy holes in the dense network. This is because the edge weights and the edge weight differences become smaller in the dense network. The lighter the edge, the less energy would a node consumes. The less the edge weight differences, the more efficient of avoiding energy holes. Thus, the longer lifetime a node could have.

We next show that the proposed algorithms can meet the conditions for avoiding en-



(a) Simulation result of a 20-node WSN

(b) Simulation result of a 50-node WSN

Figure 9.2: Simulation results.

ergy holes, as mentioned in Sec. 7.1. First, we show that EHAEC meets the routing construction viewpoint condition. Since lines 10-16 in Algorithm 1 define the child nodes for an internal node by using phony weights, EHAEC can balance the number of child nodes between internal nodes in tree *A*. As an example, Figure 9.3 shows the data communication routes for 50-node WSN simulation using the PRIM algorithm and EHAEC. It shows that EHAEC forms a route that results in the internal nodes having the same number of child nodes as much as possible compared with the PRIM algorithm. Next, we show that TINORESA and COMSA meet the node lifetime viewpoint condition. Since lines 5-10 in Algorithm 2 select the tired nodes and disable their abilities to relay child nodes in order to balance the energy consumption of the tired nodes, TINORESA can find

Table 9.2: Algorithms' efficiency by DIRECT in the 50-node network

	DIRECT	PRIM	EHAEC	TINORESA	COMSA
1%	1	3.7	4.8	6.5	6.0
50%	1	3.3	3.4	3.2	3.1
100%	1	0.6	0.6	0.5	0.5

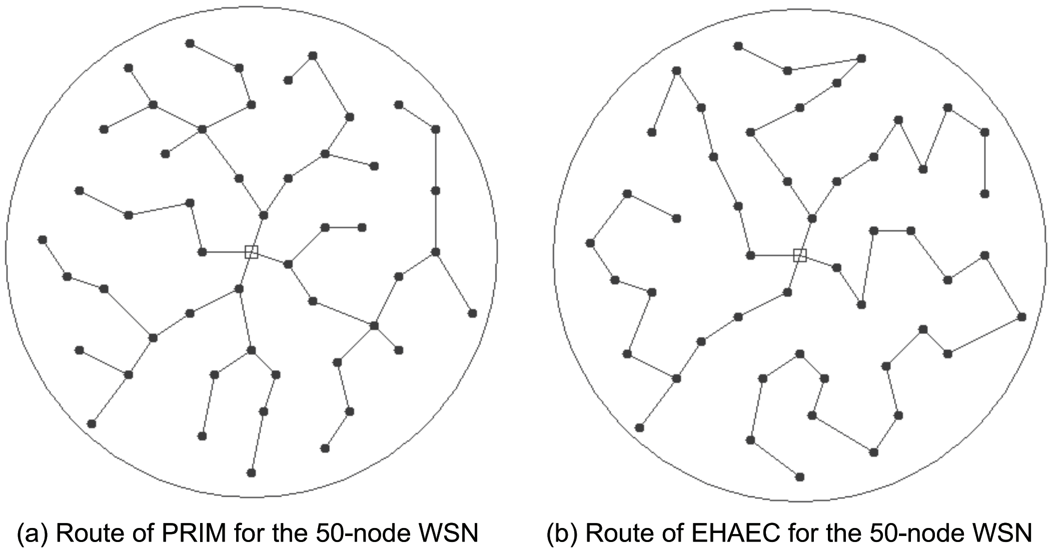

(a) Route of PRIM for the 50-node WSN
(b) Route of EHAEC for the 50-node WSN

Figure 9.3: Topology comparison of PRIM and EHAEC in the 50-node WSN.

several switching routes to balance the node lifetimes. Lines 11-14 of Algorithm 3 compute the complementary energy consumption tree of EHAEC, thereby enabling COMSA to balance the node lifetimes. For example, the simulation results in Fig. 9.2 show that TINORESA and COMSA both equalize the node lifetimes as much as possible.

9.2 Simulations of Fault Tolerant Algorithms

In the simulations for evaluate fault tolerant algorithms, we consider fault tolerance is more important than energy efficiency. We simulated the 50-node WSN configured as shown in Fig. 9.1(2). The lifetime of a battery-powered WSN is defined as the number of communication rounds till $\lambda\%$ of the sensor nodes stop operating, where λ is specified by the system designer [50]. For the consideration of fault tolerance, we assumed that the WSN remained functional as long 50% of the nodes were operating.

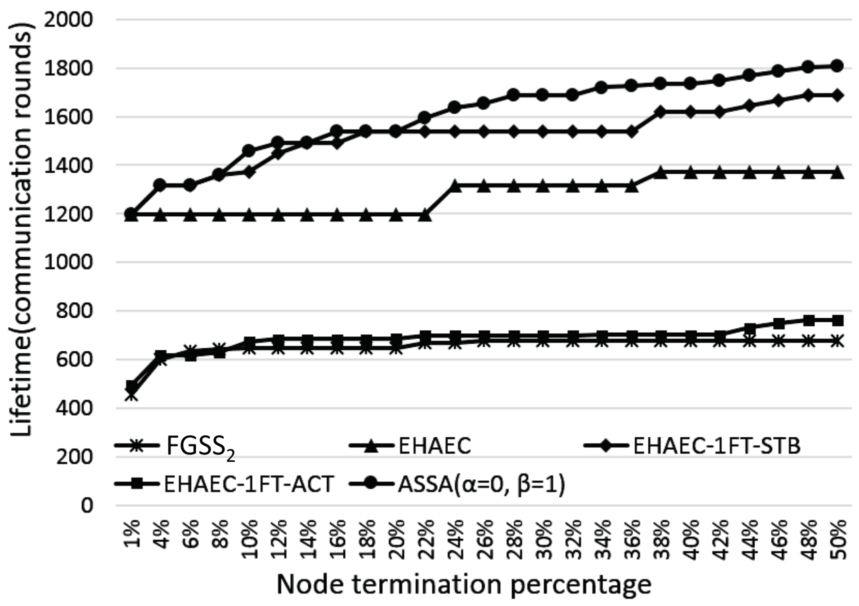


Figure 9.4: Simulated lifetime for the second situation (fault tolerance is more important than energy efficiency).

Table 9.3: Efficiency when fault tolerance is more important than energy efficiency.

	EHAEC	EHAEC-1FT-STB	EHAEC-1FT-ACT	ASSA	$FGSS_2$
1%	1	1	0.41	1	0.38
25%	1	1.17	0.53	1.25	0.51
50%	1	1.23	0.55	1.32	0.37

9.2.1 Fault-Tolerance-First Evaluation

Considering fault tolerance is more important than energy efficiency, we assumed that a node had stopped operating when there were no possible edges for it to transmit its data to the base station, even if the node had not run out of energy. Consider the example in Fig. 8.2(2). Although the route guarantees 2-connectivity, if nodes *b* and *d* have stopped, nodes *e* and *f* are also considered to have stopped because they cannot transmit data to the base station. For the route generated by EHAEC, we compared the ASSA and EHAEC-1FT with the standby and active fault tolerances using EHAEC (which does not consider fault tolerance). We used $\alpha = 0$ and $\beta = 1$ in this simulation, considering that we wanted the best performance in terms of energy efficiency. We also compared these results with $FGSS_k$. Although $FGSS_k$ can find a k-connected spanning subgraph, we assumed that it maintained 2-connectivity in the simulations so that we could compare it with our algorithms. Therefore, $FGSS_k$ is $FGSS_2$ in these simulations. Figure 9.4 shows the simulation results. Table 9.3 shows the corresponding energy efficiencies. With standby fault toler-

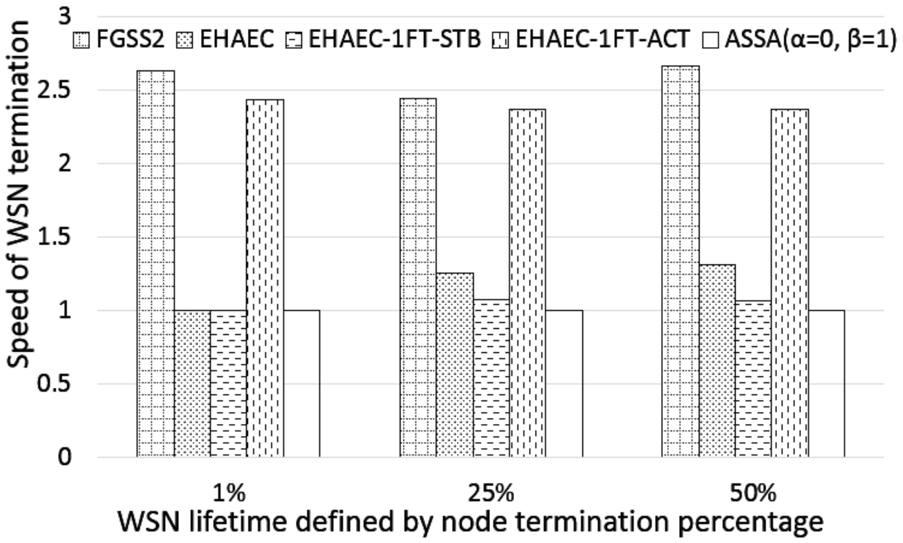


Figure 9.5: Simulated fault tolerant effect when fault tolerance is more important than energy efficiency.

ance, EHAEC-1FT was more efficient than EHAEC, and with active fault tolerance it was less efficient. This is because the communication tree used with the active fault tolerance was larger than that used with the standby fault tolerance. Therefore, it consumed more energy. From the fault tolerance perspective, EHAEC cannot tolerate node failure whereas EHAEC-1FT can tolerate one node failure using standby and active fault tolerance. Moreover, the ASSA was more efficient than the other three methods, because the batteries of all the nodes were used and because it can tolerate all node failures at any time. EHAEC-1FT and ASSA are more efficient than $FGSS_2$. However, the results of $FGSS_2$ and EHAEC-1FT in active mode are almost the same. This is because $FGSS_2$ generates a 2-connected spanning tree based on Kruskal's algorithm, and the redundant routes are always applied during wireless communications. Next, we investigated the fault tolerant effects of the algorithms by comparing the WSN termination speeds in Fig. 9.5. The speed of the WSN termination is defined as $speed = \frac{lifetime_{longest}}{lifetime_{self}}$, where $lifetime_{longest}$ is the longest lifetime of the five simulated algorithms under the current WSN lifetime definition (i.e., the length of time until $\lambda\%$ of sensor nodes stop operating), and $lifetime_{self}$ is the lifetime of a WSN using one of the algorithms. A quicker speed corresponds to a worse fault tolerance. Therefore, Fig. 9.5 shows that the ASSA performed the best and $FGSS_2$ was the worst.

We also compared the energy efficiency and failure times using different values of α and β: $\alpha = 1$ and $\beta = 0$, $\alpha = 0.5$ and $\beta = 0.5$, and $\alpha = 0$ and $\beta = 1$. Figures 9.6 and 9.7 show the results. In Fig. 9.6, there were no significant differences when using different values of α and β (because the ASSA finds spare nodes for the route generated

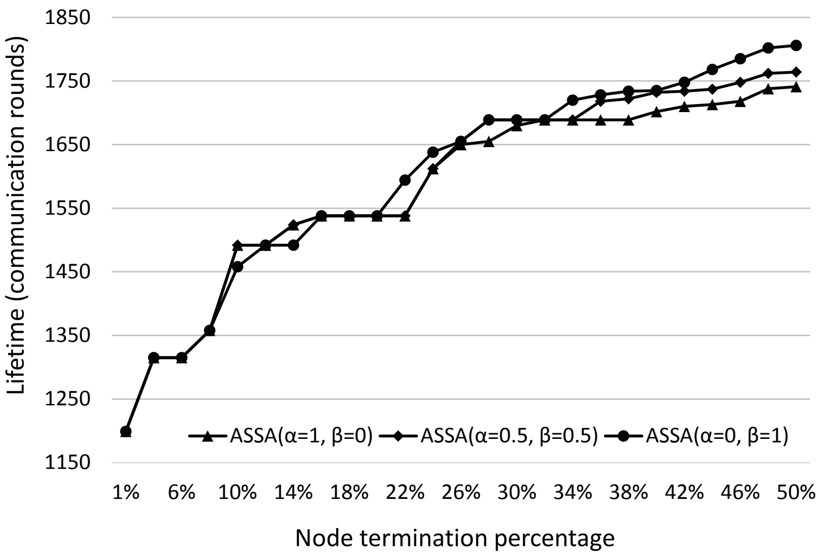


Figure 9.6: Simulated lifetime for the ASSA with different values of α and β.

by EHAEC, which is originally energy efficient). However, the ASSA results appears to be more energy efficient for larger β. There were less node failures when α was larger (Fig. 9.7). The result agrees with idea behind the ASSA.

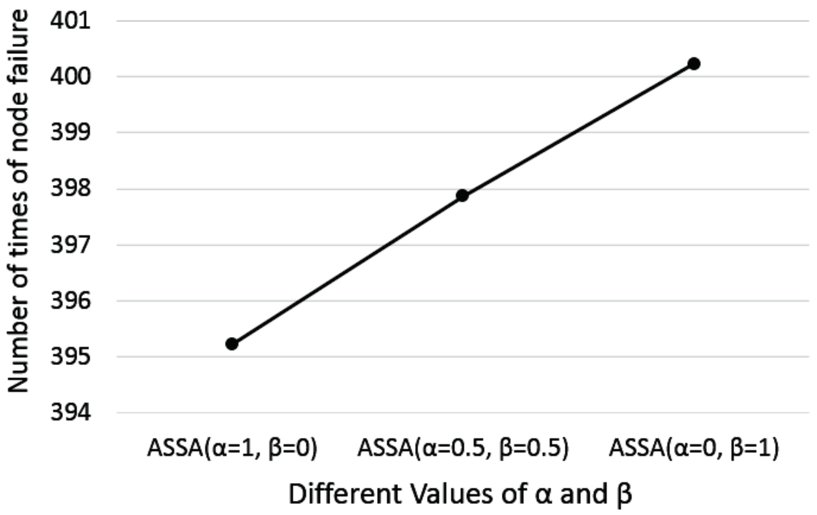


Figure 9.7: Simulated number of node failures for the ASSA with different values of α and β.

Chapter 10

Summary and Conclusions

In this doctoral thesis, we raised a maintenance problem in wireless sensor networks that battery powered sensor nodes in harsh environment should be operated in a prolong period in order to avoid changing their batteries. Since battery energy is mainly consumed by wireless communication, sensors in a node, and CPUs where tasks that are used to control sensors in a WSN, we attempted to reduce energies consumed by those aspects. In the way toward to the goal, we separately investigated the sensor and task scheduling problem as well as the wireless communication problem, both impacting the lifetime of WSNs.

10.1 Sensor and Task Scheduling

From the perspective of sensor and task scheduling, We first presented an experiment showing the most significant factor affecting the battery life in a sensor node is peak power consumption, and when peak power consumption is high, battery voltage drops quickly, and the sensor stops working even though some useful charge remains in the battery. Our proposal extends battery life by scheduling sensors' execution time that is able to reduce peak power consumption as much as possible under a deadline constraint. Furthermore, of the three algorithms we devised, we found BFSA to be the most efficient for scheduling sensor IO to extend battery life when we tested it in a simulator that we designed. The simulations predict the battery life of each algorithm in order to evaluate its effectiveness. The simulation results showed that by comparing with executing sensors simultaneously, the three algorithms dramatically can extend battery life, BFSA can extend battery life approximately three time as long as simultaneous sensor activation, thus, the most effective algorithm is BFSA. Moreover, in order to reduce the energy consumed by tasks, we proposed two task scheduling algorithms, DVFS-PTEA and ETS-PTEA, activate and deactivate sensors with dynamic voltage and frequency scaling or with execution time scaling in order to avoid wasting energy in the CPU. We showed the effectiveness of these two algorithms by comparing their energy consumptions with that of continuous task execution in the cases of the three sensor scheduling algorithms. The evaluation results showed that ETS-PTEA is the most energy efficient solution for scheduling tasks, DVFS-PTEA is the second, and continuous task execution is the third.

10.2 Energy Efficient Wireless Communication

From the other perspective of wireless communication, we investigated the energy efficiency and fault tolerance of WSNs, given that the communication distances and energy hole problem are critical to the battery life of the sensor nodes, which affects the network's lifetime. We first presented a new routing algorithm EHAEC based on the PRIM algorithm for achieving energy efficient wireless communications by minimizing the energy consumption of sending and relaying data. EHAEC can avoid energy holes to the maximum extent possible without network routing reconstruction and is well suited for WSNs for which the lifetime is defined by a high node termination percentage. However, for WSNs for which the lifetime is defined by a low node termination percentage, the energy hole problem should be solved by network routing reconstruction to increase the lifetime of short lifetime nodes and balance the lifetime of each node. For that purpose, we also proposed two route switching algorithms TINORESA and COMSA to balance the energy consumption of each node in the network in order to solve the energy hole problem. TINORESA finds several switching routes to solve the problem but has high route switching overhead. In contrast, COMSA only finds one switching route and has low route switching overhead. To evaluate the algorithms' effectiveness, simulations were performed to predict the lifetime of WSNs for each of them. The simulation results showed that by avoiding energy holes, our proposed EHAEC, TINORESA, and COMSA algorithms can extend the lifetime to more than 3 to 6 times that for direct data transmission when 1% of the nodes are terminated in a 20-node network and a 50-node network. Moreover, considering that a WSN may have unreliable sensor nodes, we developed two proactive fault tolerant routing algorithms. The EHAEC-1FT algorithm is based on EHAEC, and constructs two-connectivity routes that achieve one-node failure tolerance in WSNs. We also extended EHAEC-1FT to X-nFT for n-node failure tolerance using an arbitrary spanning tree routing algorithm X. We developed the ASSA to select active spare nodes in the network. By varying the impact factors, α and β, the ASSA can select active spare nodes that make the route more energy efficient or more fault tolerant. Further simulation results showed that the proposed fault tolerant algorithms were more effective than $FGSS_k$. EHAEC-1FT with standby fault tolerance more effectively extended the WSN's lifetime than EHAEC, and EHAEC-1FT with active fault tolerance was less effective than EHAEC. EHAEC-1FT was more effective than EHAEC in terms of the fault tolerance, because it can tolerate the failure of one node. Moreover, our simulation results showed that the ASSA was more efficient than EHAEC-1FT, that large values of α produce a highly fault tolerant topology, and that large values of β increase the energy efficiency.

10.3 WSN Applications Using Proposed Methods

In this section, we introduce some WSN applications that our proposed methods are suited for. WSN systems have been applied in various aspects such as enhanced safety and security, smart agriculture and intelligent buildings etc.. Since our methods were proposed for general purpose, they can be utilized in many different WSN applications such as the mentioned aspects.

From the viewpoint of safety and security of our society, recently, due to a number of catastrophic failures, many countries have been focusing on the maintenance of their infrastructures, such as bridges, tunnels, or even nuclear power plants. Taking bridges as an example, in California of United States, 13% of the 23,000 bridges have been deemed structurally deficient, while 12% of the US's 600,000 bridges share the same rating. And the situation is same in Japan. Japan has approximately 157,000 bridges of 15 m or more in length, and the number of bridges that have been in use for 50 or more years was approximately 15,000 in 2011 and will be 44,000 and 84,000 in 2021 and 2013. Therefore, structural health monitoring carried out by WSN applications is required for bridge inspections. Sensor nodes deployed in bridges require longer lifetime since maintain sensor nodes in places such as bridges and tunnels is not a simple task. In bridge monitoring sensor nodes, various sensors such as vibration sensors, pressure sensors and acceleration sensors are embedded in a node, hence, sensor and task scheduling algorithms can help to extend the battery life. Moreover, due to bridges should be covered by many sensor nodes while monitoring, the proposed topology management algorithms can assist the WSN to find an energy efficient route.

Moreover, in the applications of agricultures, it is important for farm owners to manage cultivation of plants in order to get the exact conditions in which plants are growing in a comfort environment. Such a WSN application is required to control conditions in the farmland and monitor high performance of plants. Taking advantage of sensing data of best performance can help people to create proper climate and to use proper amount of fertilizers, since a slight change in climate and inappropriate amount of fertilizers can affect the final outcome. This kind of WSN applications utilized in farms usually contain plenty of sensor nodes, and the sensor nodes contain sensors such as humidity, temperature, and light sensors in order to detect the risk of frost, plant diseases and watering requirement based on the sensing data. Without a question, our proposed energy efficient approaches are also effective for the agriculture monitoring application in extending the WSN lifetime. Our proposed methods not only fit for the applications we raised, but also fit other applications such as smart homes, intelligent buildings, earthquake monitoring, radiation prevention, natural environment protection, since the methods are proposed for general purpose.

Acknowledgments

First and foremost I would like to thank my adviser Professor Yukikazu Nakamoto. His passion and thoroughness made this thesis possible, he always actively working to improve my research. I will always remember Professor Nakamoto's kindness not only did Professor Nakamoto teach me a wealth of knowledge and help me to approach my goal in academic and career, but also he gave me a great many advises in my studying abroad life. His commitment will always inspire me.

I would like to extend my gratitude towards Professor Danny Fernandes for his English support to me, and also Professor Danny Fernandes gave me plenty advises in my study and everyday life.

I would also like to thank Mr Masayoshi Kai, Makoto Iwata and Koutaro Yamamura in NEC, who shared valuable information with me in the joint research, and gave me a lot of supports and advice in my internship in NEC.

Special thanks to all Professors in Graduate School of Applied informatics and many thanks to the Nakamoto Laboratory members for all advice, support and discussions.

Part of this work was supported by New Energy and Industrial Technology Development Organization (NEDO), Japan. This work also is a joint research with NEC. I would like to acknowledge the support of NEDO and NEC.

This thesis is based on [51] (©IEICE 2013) and [52] (©IEICE 2014). We would like to thank IEICE for the usage.

Last but not least, I would like to thank my family members for all cares and supports during my master and doctoral course in Japan.

Bibliography

[1] M. Hermann, T. Pentek, and B. Otto, “Design principles for industrie 4.0 scenarios: A literature review,” 2015.

[2] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*. 3rd ed., Springer, 2011.

[3] A. Burns and A. Wellings, *Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real-Time POSIX*, 4th ed. Addison-Wesley Educational Publishers Inc, 2009.

[4] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*, 1st ed. Edward Ashford Lee & Sanjit Arunkumar Seshia, 2015.

[5] C. L. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard-real-time environment,” *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, January 1973.

[6] R. Graybill and R. Melhem, *Power Aware Computing*. Springer Science & Business Media, 2013.

[7] T. D. Burd and R. W. Brodersen, “Energy efficient cmos microprocessor design,” in *Proceedings of the 28th IEEE Annual Hawaii International Conference on System Sciences*, vol. 1, pp. 288–297, January 1995.

[8] T. Okuma, H. Yasuura, and T. Ishihara, “Software energy reduction techniques for variable voltage processors,” *IEEE Design & Test of Computers*, vol. 18, no. 2, pp. 31–41, March 2001.

[9] P. Pillai and K. G. Shin, “Real-time dynamic voltage scaling for low-power embedded operating systems,” in *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5, pp. 89–102, October 2001.

[10] A. Andrei, M. T. Schmitz, P. Eles, Z. Peng, and B. M. A. Hashimi, “Quasi-static voltage scaling for energy minimization with time constraints,” in *Proceedings of*

the Design, Automation and Test in Europe Conference and Exhibition, pp. 514–519, March 2005.

[11] A. Andrei, “Energy efficient and predictable design of real-time embedded systems,” Ph.D. dissertation, Dept. of Computer and Information Science, Linkoping University, October 2007.

[12] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. M. Al-Hashimi, “Overhead-conscious voltage selection for dynamic and leakage energy reduction of time-constrained systems,” *IET Computers and Digital Techniques*, vol. 152, no. 1, pp. 28–38, January 2005.

[13] O. Jovanovic, *Low Power Software for Multiprocessor Systems*. VDM Verlag, 2008.

[14] A. Krause, R. Rajagopal, A. Gupta, and C. Guestrin, “Simultaneous placement and scheduling of sensors,” in *Proceedings of the 2009 IEEE International Conference on Information Processing in Sensor Networks*, pp. 181–192, 2009.

[15] M. R. Jongerden and B. R. Haverkort, “Battery modeling,” Centre for Telematics and Information Technology, University of Twente, Tech. Rep., January 2008.

[16] D. Linden and T. B. Reddy, *Handbook of Batteries*. McGraw-Hill, Inc, 2002.

[17] M. Doyle, T. F. Fuller, and J. Newman, “Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell,” *Journal of the Electrochemical Society*, vol. 140, no. 6, pp. 1526–1533, 1993.

[18] D. N. Rakhmatov and S. B. Vrudhula, “An analytical high-level battery model for use in energy management of portable electronic systems,” in *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, pp. 488–493, 2001.

[19] C. F. Chiasserini and R. R. Rao, “Pulsed battery discharge in communication devices,” in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 88–95, 1999.

[20] M. Iwata, M. Kai, and H. Shimazu, “Development of the energy management platform “green tap (1)”,” in *Proceedings of the 73rd annual conference of Information Processing Society of Japan*, pp. 4–381–4–382, March 2011, in Japanese.

[21] K. Yamamura, K. Ishida, M. Iwata, M. Kai, and H. Shimazu, "Development of the energy management platform "green tap (2)"," development of long-lived wireless environmental sensor that can work in expert year," in *Proceedings of the 73rd annual conference of Information Processing Society of Japan*, pp. 4–383–4–384, March 2011, in Japanese.

[22] T. L. Martin and D. P. Siewiorek, "Non-ideal battery properties and low power operation in wearable computing," in *Proceedings of the 3rd IEEE International Symposium on Wearable Computers*, pp. 101–106, 1999.

[23] A. Iwasa, M. Iwata, M. Kai, and H. Shimazu, "Development of energy management platform "greentap" (3)," in *Proceedings of the 73rd annual conference of Information Processing Soceity of Japan*, pp. 4–385–4–386, March 2011, in Japanese.

[24] H. Kopetz, *Real-time systems: design principles for distributed embedded applications*, 2nd ed. Springer Science & Business Media, 2011.

[25] M. Ding, X. Cheng, and G. Xue, "Aggregation tree construction in sensor networks," in *Proceedings of the 58th IEEE Vehicular Technology Conference*, vol. 4, pp. 2168–2172, October 2003.

[26] N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," in *Proceedings of the International Conference on Information Technology: Coding and Computing*, vol. 2, pp. 8–13, April 2005.

[27] S. Olariu and I. Stojmenovic, "Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting," in *Proceedings of the IEEE Conference on Computer Communications*, pp. 1–12, April 2006.

[28] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2007.

[29] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Hawaii International Conference on System Sciences*, vol. 2, 2000.

[30] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 174–185, August 1999.

[31] Y. Yao and J. Gehrke, “The cougar approach to in-network query processing in sensor networks,” *ACM Sigmod Record*, vol. 31, no. 3, pp. 9–18, September 2002.

[32] S. Lindsey and C. S. Raghavendra, “Pegasis: Power-efficient gathering in sensor information systems,” in *Proceedings of the IEEE Aerospace Conference*, vol. 3, pp. 3–1125–3–1130, 2002.

[33] T. H. Cormen, C. E. Leiserson, R. L. Rovest, and S. C, *Introduction to algorithms*, 3rd ed. MIT press, 2009.

[34] J. Lian, L. Chen, K. Naik, T. Otzu, and G. Agnew, “Modeling and enhancing data capacity in wireless sensor networks,” *IEEE Monograph on Sensor Network Operations*, pp. 91–138, 2004.

[35] X. Wu, G. Chen, and S. K. Das, “Avoiding energy holes in wireless sensor networks with nonuniform node distribution,” *IEEE Trans on Parallel and Distributed Systems*, vol. 19, no. 5, pp. 710–720, May 2008.

[36] X. Liu, “A survey on clustering routing protocols in wireless sensor networks,” *Sensors*, vol. 12, no. 8, pp. 11 113–11 153, 2012.

[37] S. Tanessakulwattana, C. Pornavalai, G. Chakraborty, and S. Naik, “Optimal multipath energy-aware routing protocol for wireless sensor networks,” in *Proceedings of the 9th IEEE International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, pp. 1–4, 2012.

[38] J. W. Jung and M. A. Ingram, “Residual-energy-activated cooperative transmission (react) to aoid the energy hole,” in *Proceedings of the IEEE International Conference on Communications Workshop*, pp. 1–5, 2010.

[39] M. Younis, I. F. Senturk, K. Akkaya, S. Lee, and F. Senel, “Topology management techniques for tolerating node failures in wireless sensor networks: A surve,” *Computer Networks*, vol. 58, pp. 254–283, 2014.

[40] J. C. Hou, N. Li, and I. Stojmenovic, *Topology Construction and Maintenance in Wireless Sensor Networks, Handbook of sensor networks: algorithms and architectures*. John Wiley & Sons, 2005.

[41] N. Li and J. C. Hou, “Flss: a fault-tolerant topology control algorithm for wireless networks,” in *Proceedings of the 10th annual international conference on Mobile computing and networking*, pp. 275–286, 2004.

[42] X. Y. Li, P. J. Wan, Y. Wang, and C. W. Yi, "Fault tolerant deployment and topology control in wireless networks," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pp. 117–128, 2003.

[43] M. Jorgic, M. Hauspie, D. Simplot-Ryl, and I. Stojmenovic, "Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks," in *Proceedings of the 3rd IFIP Mediterranean Ad Hoc Networking Workshop*, pp. 360–371, 2004.

[44] A. Kashyap, S. Khuller, and M. A. Shayman, "Relay placement for higher order connectivity in wireless sensor networks," in *Proceedings of the IEEE International Conference on Computer Communications*, vol. 1, 2006.

[45] W. Zhang, G. Xue, and S. Misra, "Fault-tolerant node placement in wireless sensor networks: problems and algorithms," in *Proceedings of the IEEE International Conference on Computer Communications*, pp. 1649–1657, 2007.

[46] Z. Yun, X. Bai, D. Xuan, T. H. Lai, and W. Jia, "Optimal deployment patterns for full coverage and connectivity wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 3, pp. 934–947, 2010.

[47] K. Vaidya and M. Younis, "Efficient failure recovery in wireless sensor networks through active spare designation," in *Proceedings of the 6th IEEE International conference on Distributed Computing in Sensor Systems Workshops*, pp. 1–6, 2010.

[48] K. Akkaya, A. Thimmapuram, F. Senel, and S. Uludag, "Distributed recovery of actor failures in wireless sensor and actor networks," in *Proceedings of the IEEE Wireless Communication and Networking Conference*, pp. 2480–2485, 2008.

[49] Zigbee and Alliance, "Zigbee specification faq," retrieved 4 April 2014. [Online]. Available: http://www.zigbee.org/Specifications/ZigBee/FAQ.aspx.

[50] R. Rajagopalan and P. K. Varshney, "Data aggregation techniques in sensor networks: A survey," *IEEE Communications Surveys and Tutorials*, vol. 8, no. 4, pp. 48–63, 2006.

Publications

[51] Q. Zhao, Y. Nakamoto, S. Yamada, K. Yamamura, M. Iwata, and M. Kai, "Sensor scheduling algorithms for extending battery life in a sensor node," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E96-A, no. 6, pp. 1236–1244, June 2013.

[52] Q. Zhao and Y. Nakamoto, "Algorithms for reducing communication energy and avoiding energy holes to extend lifetime of wsns," *IEICE Transactions on Information and Systems*, vol. E97-D, no. 12, pp. 2995–3006, December 2014.

[53] Q. Zhao and Y. Nakamoto, "Topology management for reducing energy consumption and tolerating failures in wireless sensor networks," *International Journal of Networking and Computing*, vol. 6, no. 1, pp. 107–123, January 2016.

[54] Q. Zhao and Y. Nakamoto, "Energy-efficient sensor and task scheduling for extending battery life in a sensor node," in *Proceedings of the 1st IEEE International Conference on Cyber-Physical Systems, Networks and Applications*, pp. 96–100, August 2013.

[55] Q. Zhao and N. Yukikazu, "Routing algorithms for preventing energy holes and improving fault tolerance in wireless sensor networks," in *Proceedings of the 2nd International Symposium on Computing and Networking*, pp. 278–283, December 2014.

[56] Q. Zhao, "Extending the lifetime of wireless sensor networks from the perspective of sensor scheduling and wireless communication," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Ph.D. Forum*, pp. 233–235, March 2015.

[57] Q. Zhao and Y. Nakamoto, "Estimating the battery consumption of data processing in a wireless sensor node," in *Proceedings of the 1st International Symposium on Computing and Networking*, pp. 484–486, December 2013.

[58] Q. Zhao and Y. Nakamoto, “Energy-efficient protocol for extending battery life in wireless sensor networks,” in *Proceedings of the 33rd IEEE International Conference on Distributed System*, pp. 268–273, July 2013.

[59] Q. Zhao, Y. Nakamoto, S. Yamada, K. Yamamura, M. Iwata, and M. Kai, “Sensor scheduling methods for efficient battery usage,” in *Proceedings of the Embedded System Symposium 2012 Japan*, pp. 216–217, October 2012, in Japanese.

[60] Q. Zhao, Y. Nakamoto, S. Yamada, K. Yamamura, M. Iwata, and M. Kai, “Scheduling sensor io for minimizing battery consumption and voltage drop in a sensor node,” in *Proceedings of the IEEE International Symposium on low-power and high-speed chips*, p. 22, April 2012.