Sleipnir: A Versatile Extremely Low Duty-Cycle Sensor Network

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Yu Gu

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor Of Philosophy

Prof. Tian He, Advisor

May, 2010

Acknowledgements

There are many people that have earned my gratitude for their contribution to my time in graduate school. First of all, I sincerely appreciate the education, guidance, encouragement, and inspiration from my advisor Professor Tian He. He is the person who brought me into research, and has been extremely helpful, understanding, and patient. I owe him my deepest gratitude. I also would like to thank Professor Zhi-Li Zhang, Professor Yongdae Kim, and Professor Nihar Jindal for being on my dissertation committee and for their immensely constructive and insightful suggestions and feedbacks.

I would also like express my sincere gratitude to Professor John Stankovic, Professor David Du, Dr. Fan Ye, Professor Jinhui Xu, Dr. Joengmin Hwang, Dr. Jaehoon Jeong, Dr. Qingquan Zhang, Ziguo Zhong, Ting Zhu, Shuo Guo, Mingen Lin, Dr. Ting Yan, Dr. Zhe Zhang, Dr. Hao Yang, Dr. Minkyong Kim, Dr. Hui Lei, Dr. Zhen Liu, Professor Gerald Sobelman, Professor Radu Stoleru and Yafeng Wu, who collaborated with me on my projects and papers. I also thank my many of my friends and colleagues, including, but not limited to the following: Xiao Zhao, Ke Sheng, Jing Jin, Jie Li, Lei Xiao, Ye Yuan, Cong Liu, Nan Li, Shuang Li, Yu Jin, Qiang Fu and Dingcheng Li.

I would like to show my deepest thanks to my family. I sincerely thank my wife, Likun Zheng for her full support, patience and love. I am deeply indebted to my parents, Lipeng Gu and Yan Wang, for their love, support, and encouragement throughout my life. Also, I also like to give my sincere thanks to my aunt, Lihong Gu for her support and guidance throughout my life. I would like to thank my host parents, Richard and Bettye Glass, for their love and encouragement. Finally, I would show my greatest thanks and love to my grandfather Gang Gu, who is the greatest person ever in my heart and encouraged me to pursue a scientific career in my childhood. I am missing him in heaven and this dissertation is solely dedicated to him.

Dedication

To my grandfather Gang Gu.

Sleipnir: A Versatile Extremely Low Duty-Cycle Sensor Network

by Yu Gu

ABSTRACT

Wireless Sensor Network (WSN), is a new information paradigm based on the collaboration of a large number of self-organized sensing nodes. With the increasing demand of cyber-physical interaction, wireless sensor networks have emerged as one of key technologies for many promising applications such as assisted living, military surveillance, infrastructure protection and scientific exploration. Sensor networks are acclaimed to be low-cost, low-profile, and easy to deploy. These attractive advantages, however, imply the resources available to individual nodes are severely limited. Although it is highly possible that the constraints on computation and storage disappear along with the fast development of fabrication techniques, the energy constraint is unlikely to fade away quickly. On the other hand, many sensor network based applications require a lifetime that ranges from several months to tens of years. In order to bridge the gap between limited energy supply and long-term operation requirement of applications, we then have to build extremely low duty-cycle sensor networks where during the operation of sensor applications, sensor nodes activate very briefly and stay in a dormant state for a very long period of time. In this dissertation research, we initiate the first systematic research for low-duty-cycle sensor networks, including a generic sensing architecture, a novel data forwarding scheme for intermittently connected networks and an energy synchronized communication middleware for energy-harvesting sensor networks. The goal of this dissertation research is to provide better understanding of how to build practical and efficient extremely low duty-cycle sensor networks and support those long-term applications such as structure monitoring, traffic control and so on. We hope, toward the very end, this dissertation research can assist the transition of sensor network technology from a research concept to a general-purpose technology available for use for a wide variety of research, government and industry purposes.

Contents

List of Figures

Chapter 1

Introduction

Wireless Sensor Network (WSN), is a new information paradigm based on the collaboration of a large number of self-organized sensing nodes. With the increasing demand of ubiquitous computing, distributed control, and cyber-physical interaction, wireless sensor networks have emerged as one of key technologies for many promising applications such as assisted living, military surveillance, habitat monitoring, infrastructure protection and scientific exploration. Compared with other competing high-end technologies, sensor networks are acclaimed to be low-cost, low-profile, and easy to deploy. These attractive advantages, however, imply the resources available to individual nodes are severely limited. Although it is highly possible that the constraints on computation and storage disappear along with the fast development of fabrication techniques, the energy constraint is unlikely to fade away quickly. For instance, while state-of-the-art computational power and storage capacity double every two years, battery capacity only doubles in 35 years. For individual sensor node, which is usually powered by two AA batteries, can only last around two days if runs at full duty-cycle (work all the time). On the other hand, many sensor network based applications require a lifetime that ranges from several months to tens of years. In order to bridge the gap between limited energy supply and long-term operation requirement of applications, we then have to build extremely low duty-cycle sensor networks where during the operation of sensor applications, sensor nodes activate very briefly and stay in a dormant state for a very long period of time. The challenges of building such extremely low duty-cycle sensor

networks therefore are conserving as much energy as possible while fulfilling various specified system requirements from targeted applications.

The goal of this dissertation research is to design fundamental system components for extremely low duty-cycle sensor networks such that both energy efficiency and system performance can be achieved simultaneously. Specifically, this dissertation focuses on following three essential building components for extremely low-duty-cycle sensor networks. (i) A generic sensing architecture that effectively reduces node duty-cycle to an extremely low level and is capable of meeting various requirements on sensing qualities. (ii) The data communication process in extremely low duty-cycle sensor networks is significantly different from traditional always-awake networks, therefore a new efficient data forwarding scheme is introduced. (iii) For sensor networks that harvest ambient energy as a complementary power supply, an energy synchronized communication scheme is discussed for dynamically adjusting node duty-cycles while minimizing network delay under varying energy supply over time.

First, for the sensing coverage component, we aim at designing a generic sensing coverage architecture that could be easily integrated by sensor network applications. Specifically, we introduce a novel two-level scheduling scheme that separates system behavior specifications from node duty-cycle scheduling, which significantly improves flexibility and system performance than existing state-of-the-art solutions.

Secondly, for the data forwarding component, we aim at tackling the challenge of delivering data messages in a mostly sleep network, which exhibits intermittent, time-dependent connectivities. In order to overcome this challenge, we design a novel dynamic switching-based forwarding technique that optimizes data delivery ratio, data delivery delay and energy consumption.

Thirdly, for the energy synchronization component for energy-harvesting sensor networks, the extracted external energy may be varied significantly over time and each sensor node needs to balance its duty-cycle with current energy supply. We introduce a constant time complexity solution for adjusting node duty-cycle with varying energy supply over time such that the communication delay at each sensor node is minimized.

This dissertation research is the first comprehensive study and design for extremely low duty-cycle sensor networks. The results of this dissertation research serve as a general guideline on how to build practical and efficient extremely low duty-cycle sensor

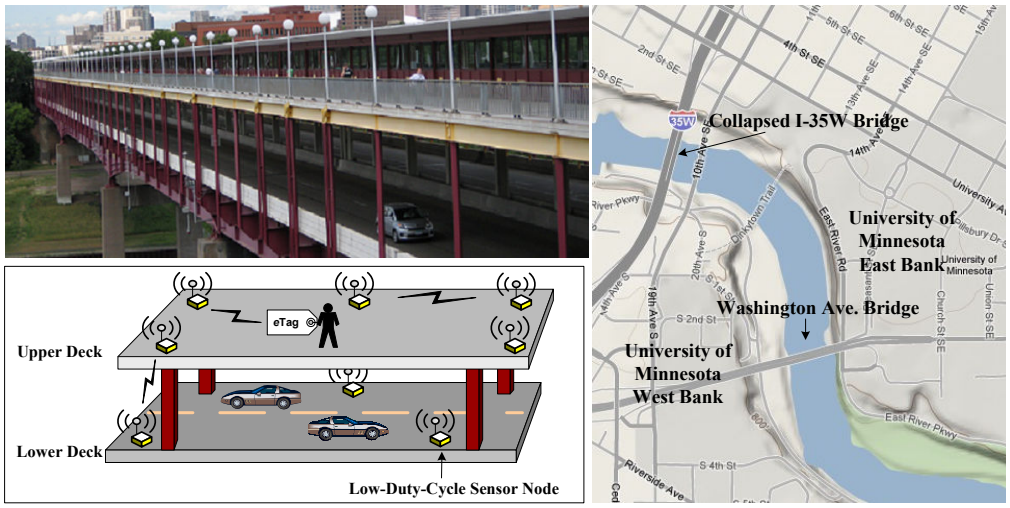


Figure 1.1: A Low-Duty-Cycle Network for Bridge Structure Monitoring

networks. Many long-term sensor network applications, such as bridge structure monitoring shown in Figure 1.1, can be supported by our design, leading to high efficiency in both performance and energy conservation. By doing so, we hope, toward the very end, this dissertation research can assist the transition of sensor network technology from a research concept to a general-purpose technology available for use for a wide variety of research, government and industry purposes.

The rest of this dissertation is organized as follows:

- Chapter 2 introduces an unified asymmetric sensing coverage architecture design for building extremely low-duty-cycle sensor networks.
- Chapter 3 describes a novel dynamic switching-based data forwarding scheme for the unicast service in low-duty-cycle sensor networks.
- Chapter 4 discusses a low-cost, energy synchronized communication middleware for low-duty-cycle, energy-harvesting sensor networks.
- Chapter 5 concludes this dissertation and presents future work for building highly efficient and reliable extremely low-duty-cycle sensor networks.

Chapter 2

Asymmetric Sensing Architecture for Wireless Sensor Networks

As a key approach to achieve energy efficiency in sensor networks, sensing coverage has been studied extensively in the literature. Researchers have designed many coverage protocols to provide various kinds of service guarantees on the network lifetime, coverage ratio and detection delay. While these protocols are effective, they are not flexible enough to meet multiple design goals simultaneously. In this chapter, we propose a Unified Sensing Coverage Architecture, called uSense, which features three novel ideas: *Asymmetric Architecture*, *Generic Switching* and *Global Scheduling*. We propose asymmetric architecture based on the conceptual separation of switching from scheduling. Switching is efficiently supported in sensor nodes, while scheduling is done in a separated computational entity, where multiple scheduling algorithms are supported. As an instance, we propose a two-level global coverage algorithm, called uScan. At the first level, coverage is scheduled to activate different portions of an area. We propose an optimal scheduling algorithm to minimize area breach. At the second level, sets of nodes are selected to cover active portions. Importantly, we show the feasibility to obtain optimal set-cover results in linear time if the layout of areas satisfies certain conditions.

Our evaluation is comprehensive: First, we provide theoretical analysis on the system lifetime and the detection performance (for both static and mobile targets) in terms

of delay and worst-case area breach. Second, we have implemented the uSense architecture and the uScan algorithm on a physical testbed, consisting of 30 MicaZ motes. Extensive empirical results are collected. Third, we implement several state-of-the-art solutions and compare them with our design in a 10,000-node simulator. Fourth, we also investigate the robustness of our design in terms of node density, time synchronization and localization error. All these results indicate that uSense is a promising architecture to support flexible and efficient coverage in sensor networks.

2.1 Introduction

Wireless Sensor Networks (WSNs), consisting of thousands of low-cost sensor nodes, have been used in many application domains such as military surveillance [1], habitat monitoring [2] and scientific exploration. Limited power supplies and difficulties in harvesting ambient energy make energy conservation a critical issue to address. Energy-efficient sensing coverage extends system lifetime by leveraging on the redundant deployment of sensor nodes. Within a couple of years, sensing coverage has become a well studied subject which provides either full coverage in both time and space [3, 4, 5, 6, 7, 8], coverage with guaranteed delay and connectivity [9, 10, 11, 12, 13, 14, 15, 16], or guaranteed target detection within a certain stealth distance [17, 18]. These algorithms are designed to be well distributed and localized, providing solid performance gains with a certain guarantee on service (e.g., a bounded delay in detection or a guaranteed lifttime). While the state-of-the-art is encouraging, we believe there are some aspects that need further investigation. First, currently different sensing coverage algorithms focus on different service guarantees (e.g., coverage vs. detection delay). Any single design is not general enough to meet a wide range of sensing requirements under different operating scenarios. Second, in most algorithms, extending system lifetime is achieved essentially through coordination among neighboring nodes. The local node density, therefore, imposes a theoretical upper bound on the system lifetime, if a continuous sensing coverage or a partial coverage is required. Such a bound can be surpassed through global scheduling. However, the overhead of global scheduling would increase significantly if the coordination among the nodes goes beyond the neighborhood.

To address these two issues simultaneously, in this chapter, we introduce a new sensing architecture, called uSense, which features three novel ideas: *Asymmetric Architecture*, *Generic Switching* and *Global Scheduling*. The key concept of uSense is the decoupling of sensing coverage into two separated functions: *scheduling* and *switching*. The former calculates the parameters of a working schedule for individual nodes, while the latter turns on/off the sensors according to the scheduling parameters. We employ an asymmetric architecture to improve the flexibility and extensibility of the design. Switching is a lightweight generic algorithm, taking several parameters as inputs. With these parameters, it can faithfully execute the sleep/awake schedule of individual nodes decided by *any* existing coverage algorithm. Switching must be supported in the sensor nodes, since a node has to be on to provide coverage and has to be off to save energy. On the other hand, it is not absolutely necessary to implement the scheduling within constrained sensor nodes. We opt to support scheduling on a powerful computational entity, which can be implemented at the second-tier nodes (e.g., Intel Stargate used in TENET [19] and ExScale [1]), or on an outside server, or on a cluster of servers to avoid single point of failure.

Asymmetric design is now considered as a promising guiding principle for sensor networks. By decoupling the scheduling function and implementing it outside the network core, we can achieve *efficiency* and *performance* simultaneously. This is because firstly, with fewer functions sharing the limited resources on a node, we can build the switching functions, the ones that must be embedded into individual sensor node, in a less stringent design space. Secondly, we can design and implement the scheduling functions outside of sensor networks, free of resource constrains inherent in the sensor nodes, therefore, will be more sophisticated and powerful, leading to a significantly improved overall performance.

Before describing the uSense design in detail, we identify the objectives and intellectual contributions of this work as follows:

- **Asymmetric Sensing Architecture:** The asymmetric architecture enables us to design sophisticated coverage algorithms in an unconstrained design space and represent such intelligence with a lightweight algorithm implemented in sensor nodes.

- **Generic Switching Algorithm:** To the best of our knowledge, we propose here the first generic and lightweight switching algorithm that is suitable for sensor nodes with limited resources. We demonstrate our lightweight design at the sensor side is very effective.

- **Global Scheduling Algorithms:** We design two new global scheduling algorithms, using the concept of set-cover. Different from all previous work, we demonstrate the feasibility to identify a minimum cover set in linear time when the coverage area is a continuous curve.

- **System Implementation and Extensive Evaluation:** We have invested significant amount of effort to evaluate our design. We have implemented and evaluated the design on the Berkeley TinyOS/Mote platform, using 30 MicaZ motes. We also provide a 10,000-node large scale simulation to (i) compare with state-of-art solutions, lower-bonds and upper-bounds, and to (ii) investigate the performance in terms of energy, detect delay and worse case breach (WCB), and to (iii) study sensitivity of the design in terms of network density, time synchronization and localization errors.

The rest of this chapter is organized as follows. Section 2.2 introduces the overarching architecture of uSense. Section 2.3 describes the design of uScan, a two-level global scheduling algorithm. Section 2.4 provides an analytic study of the proposed scheduling algorithms. Section 2.5 describes our system implementation and provides evaluation on the TinyOS/Mote platform. The results of the 10,000-node simulation are presented in two separated sections: Section 2.6 compares the performance of uSense and uScan with state-of-the-art solutions, and Section 2.7 investigates system performance and sensitivity in detail. Section 2.8 discusses the related work. Section 2.9 concludes the chapter.

2.2 uSense Architecture

One of key new ideas of this work is the conceptual separation of *switching* from *scheduling*. In this section, we provide an overview of our asymmetric sensing architecture.

2.2.1 Motivation

Our work is aiming at flexibility and efficiency in sensing coverage. It is often the case that a sensor network needs to support multiple operating scenarios. For example, a military surveillance network could be required to provide full coverage during a red alert (a spatial coverage requirement), however, it may allow a certain detection delay (a temporal coverage requirement) on other occasions to aggressively conserve energy. Two algorithms ([7] and [9]) have been successfully designed to meet these two design goals. However, neither of them, unfortunately, is flexible enough to meet both requirements. Evidently, with these two algorithms, we can achieve both functionalities by downloading two separate program images and switching between them (as supported by Deluge [20]). Clearly, separate images introduce excessive overhead in terms of communication bandwidth, energy and storage, putting flexibility and efficiency at odds with each other.

We observe that the wakeup/sleep schedules of individual sensor node can be described by two parameters, which are independent of the methods to obtain them. Therefore, the conceptual separation of switching from scheduling becomes a natural approach to resolve the conflict between flexibility and efficiency. In the uSense architecture, changing scheduling algorithms only affects the values of the parameters, not the switching logic implemented at the nodes. Consequently, flexibility can be achieved by disseminating only a few parameters.

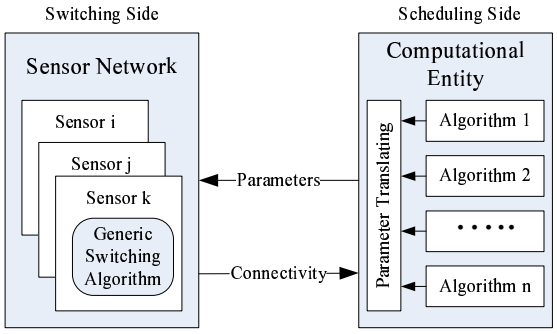


Figure 2.1: Overview of uSense Architecture

2.2.2 uSense Design

The uSense architecture decouples switching from scheduling as shown in Figure 2.1. The switching algorithm is implemented in the sensor nodes, which takes two scheduling parameters as input. The scheduling algorithm, which is implemented separately, is responsible for generating these scheduling parameters. It takes the schedules decided by various sensing coverage algorithms and translates them into the two parameters to be used by the switching algorithm. The uSense architecture requires bi-directional communication, because that (i) most scheduling algorithms need the location information of the sensor nodes in order to create a wakeup/sleep schedule for individual node, and (ii)uSense needs to disseminate the scheduling results to the nodes within the network, where the actual switching happens. Obviously, in a static sensor network, these costs are very small, compared with the amount of sensing data transmitted by the sensor network. Furthermore, various existing protocols such as SPIN [21], PSFQ [22] and EMR-Algo, DP-Algo proposed in [23] already can effectively perform energy efficient information dissemination and gathering in low power sensor networks. In the next two sections, we focus on the sensing coverage, describing the switching and scheduling algorithms, respectively.

2.2.3 Part I: Generic Switching Algorithm

The switching algorithm should be lightweight to run on resource constrained nodes and should be generic to accommodate various types of schedules. In our switching algorithm design, two parameters are used for each node, namely the schedule bits S and switching rate R.

- **Schedule bits** S is an infinite binary string in which 1 denotes the active state and 0 denotes the inactive state. Obviously, the duty cycle of a node is the percentage of 1s in S.

- **Switching rate** R defines the rate of toggling between states. For example, a switching rate of 0.5HZ requires a node to read one bit from the schedule S every 2 seconds (when consecutive bits have the same value, there is no need to change power state of the node physically). When transient energy consumption is negligible, ideally a high switch rate R leads to a small detection delay. However,

R cannot be arbitrarily small because: (i) a sensor has a warm-up delay before it can reliably sample the environment, (ii) in most detection algorithms, it is not robust enough to take just one sample to make a decision, so a sampling delay proportional to the number of samples is often introduced. These delays impose an upper-bound on the switching rate R.

Although our switching algorithm is simple and lightweight, it is powerful enough to support many types of coverage algorithms. Theoretically, when the switching rate R approaches infinity, schedule bits, as an infinite string, can precisely characterize on/off behavior generated by any coverage algorithm. As mentioned, the switching rate is finite in reality, therefore in the worst case, a node might need to extend a wake-up period by $\frac{1}{R}$ seconds to guarantee the coverage.

Regular Expression

A switching algorithm takes schedule bits S and switching rate R as inputs. Schedule bits S is formally defined as an infinite binary string. Obviously, it is practically impossible to disseminate an infinite binary string. Fortunately, the sensing coverage schedule is usually periodic, follows a certain pattern. Therefore, we can express S with a regular expression. For example, $(0010)^*$ can be used to denote a repeated off-off-active-off schedule. For a very dense network, after scheduling, the duty cycle of an individual node is usually low. In other words, the number of $1s$ in a schedule is comparatively small. In this case, we can use index values to represent the positions of active bits.

Timed Finite Automata

For a given schedule S described as a regular expression, a node builds a finite automata (FA). Clearly, a straightforward method is to use a timer at the rate of R to drive the state transitions within the FA. However, this requires a node to wake-up the processor periodically, introducing excessive energy consumption. To address this issue, we therefore build a Timed Finite Automata (TFA). In a TFA, a state transition is triggered by 01 or 10 segments in the schedule S, and the delays of transitions are the gaps between 01 or 10 segments. Figure 2.2 shows a TFA with four states. $A0$ and $A2$ are active states, while $S1$ and $S2$ are sleep states. The values on the edges denote the delays

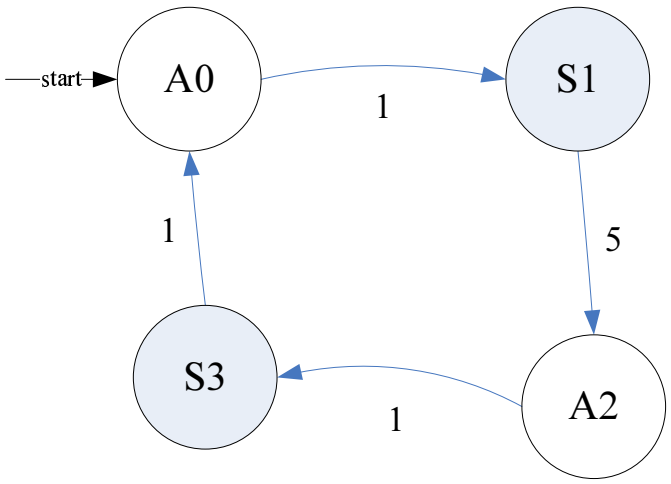


Figure 2.2: Switching with a Timed Automata

in transitions. It is easy to conclude that this TFA is used to execute the schedule $(10000010)^*$.

2.2.4 Part II: Scheduling Algorithms

As shown in Figure 2.1, a sensor scheduling algorithm is implemented separately (e.g., at the second tier). Free of resource constraints inherent in the sensor nodes, we can support a large number of coverage algorithms. In this section we focus on the generic scheduling framework, before proposing concrete scheduling algorithms in the next section.

To accommodate different sensing coverage algorithms, we need to convert the output of a coverage algorithm into two parameters understandable by the generic switching algorithm. To illustrate the idea, we use the algorithm presented in [8] as a case study.

In [8], the sensing phase of nodes is divided into rounds with equal duration T. The schedule for a node is determined by a tuple with four parameters: (T, *Ref*, T_{front}, T_{end}). As shown in Figure 2.3, *Ref* is a random time instance chosen by a node within $[0,\ T)$. T_{front} is the duration of time prior to the reference point *Ref*, and T_{end} is the duration of time after reference point *Ref*. To build a schedule bits S, we check whether a time instance $\frac{k}{R}, k \in [0, T \cdot R]$ is located within the active periods. For example, the schedule shown in Figure 2.3 can be expressed as $(00111111100)^*$.

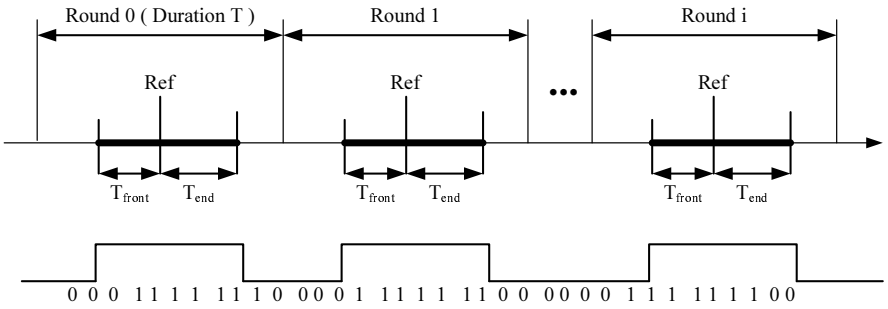


Figure 2.3: Schedule Translation

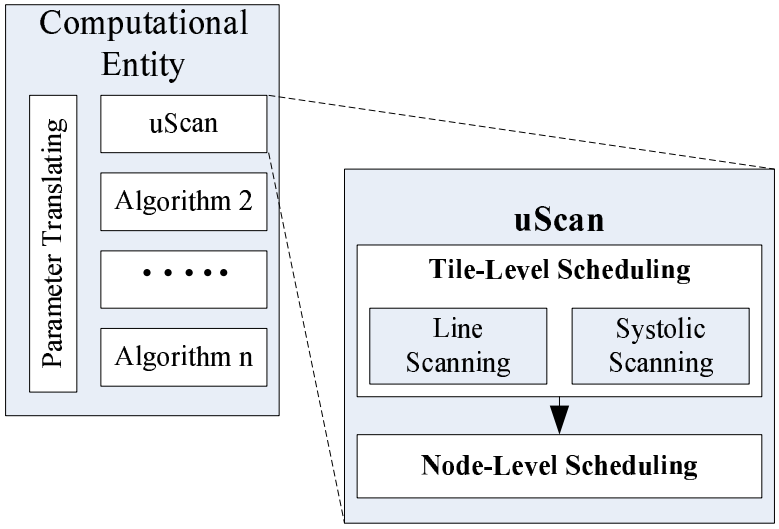


Figure 2.4: The Design of uScan

2.3 Global Scheduling Algorithms

Conceptually, uSense can support many existing coverage algorithms. Due to its asymmetric architecture, it is especially friendly to the global scheduling algorithms. Since a global scheduling allows many more nodes to activate in turn rather than the localized ones that only schedule the nodes within neighborhood, it leads to a significant energy savings. In this section, we propose a global scheduling algorithm called uScan. Figure 2.4 shows the relation between uScan and uSense. Essentially, uScan is one of sensing coverage algorithms that are supported by the uSense architecture.

The outputs of uScan are the scheduling bits S and switching rate R for individual nodes. uScan is a two-level schedule algorithm, which works as follows: Suppose we provide sensing coverage to a given area using uScan as shown in Figure 2.5. First, uScan divides the area into small regions, and decides the working schedules for these regions. This level of scheduling is conceptually independent of the deployment of the nodes. At the second-level, we assign nodes to cover the active regions at different time intervals, using a set-cover technique. By combining the first-level schedule and the set-cover assignment, we can decide the schedule bits S for individual nodes.

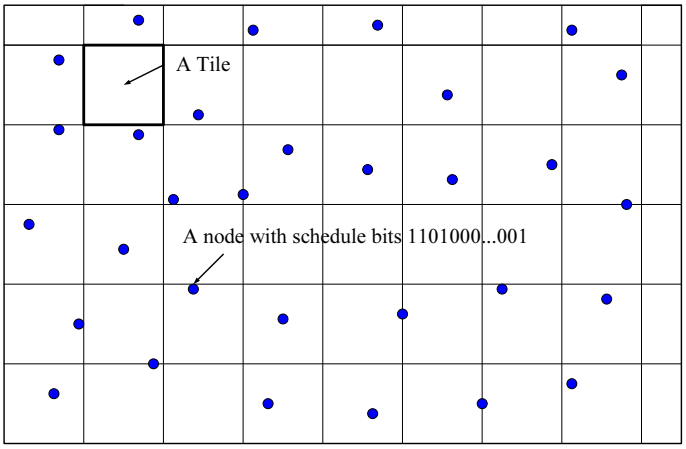


Figure 2.5: Regular Tessellations

The biggest advantage of this two-level schedule algorithm is the separation of sensing pattern from the underlying node scheduling. The application only needs to specify the desired sensing behavior on the field in the first level of scheduling, and the second level of uScan can take various specified sensing patterns as input and produce the final working schedule of each individual sensor device. The two-level schedule of the uScan provides flexibility, reusability and efficiency to the sensing component of sensor applications by freeing various sensor applications from designing their own scheduling protocols under different application requirements.

2.3.1 Assumptions

For the clarity of the protocol description in the rest of the chapter, we assume that nodes are time-synchronized and their locations are precise. We refer the sensing area

of a node as a circle with a nominal radius r centered at the location of the node. These are common assumptions for many sensor network applications [1, 2].

We will show later in evaluation section 2.7, the accuracy of time synchronization and localization do not need to be precise, and issues such as clock drift can be resolved by slightly extending the duration of active bits. In addition, our design works under irregular sensing areas as long as nodes are aware of their sensing areas [24].

2.3.2 Level I: Tile Scheduling

In uScan, we partition an area under surveillance into some small regions of the same shape, a process called tessellation. These small regions are called *tiles*, which can be regular triangles, rectangles or regular hexagons in a 2-D space. One simple example of tessellation is the rectangle-based partition, as shown in Figure 2.5. The size of tiles is set to be smaller than the minimum target size, so that a target is detected as long as a portion of a tile is covered. As a reminder, nodes within a sensor network only support a generic switching algorithm, which has neither the concept of tiles nor the partition information of the tiles. All the complex logic resides outside of sensor nodes. In this section, we describe two simple, yet effective methods for the tile-level scheduling. They differ in the energy consumption rate and the detection delay.

Line Scan

We start with a simple tile-level scheduling as shown in Figure 2.6. Instead of trying to cover all tiles, we only cover a column/row of tiles in a certain interval of time during one round of scan. The covered columns/rows are increasing or decreasing consecutively. Because only a small percentage of tiles are sensed at a specific point of time, line scan leads to a significant reduction in energy consumption, compared with full coverage [8].

Specifically, in the line-based global scanning, we introduce the concept of scanning speed v, which represents the speed of scan from one end to the other, horizontally or vertically. This scanning speed determines the maximum detection delay a network experiences. The scanning speed v can be transformed into the switch rate R. Suppose we have a rectangle-based tessellation, the length and width of a tile are L_l and L_w, respectively. For a given scan speed v, if we want to scan horizontally, the switch rate R is set to be $\frac{v}{L_l}$. Similarly, the switch rate is $\frac{v}{L_w}$, when we scan vertically. Starting from

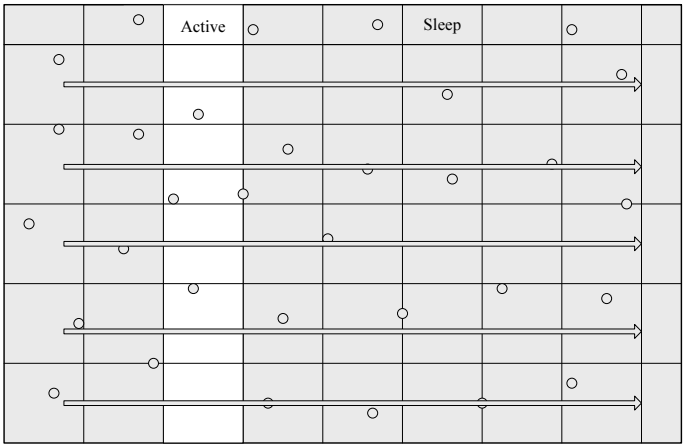


Figure 2.6: Horizontal Scan

0, we index the tiles in a row-major order. Therefore a tile with coordinates (row, col) is assigned the index of $row * col_{max} + col$ (col_{max} and row_{max} are the maximal column and row indices respectively). To cover a tile $t(i)$ with a coordinates (row, col) in a scanning round, scheduling bits S for this tile is as follows:

$$\begin{aligned} S_h(i) &= (\underbrace{000..000}_{col-1}\mathbf{1}\ \underbrace{000..000}_{col_{max}-col})^* \quad (Hscan) \\ S_v(i) &= (\underbrace{000..000}_{row-1}\mathbf{1}\ \underbrace{000..000}_{row_{max}-row})^* \quad (Vscan) \end{aligned} \tag{2.1}$$

Moreover, we can perform horizontal and vertical scans simultaneously. Both directions share the same scanning speed v. We can obtain the scheduling bits of a two-way scan by applying the bitwise OR operation on S_h and S_v obtained in Equation 2.1:

$$S(i) = S_h(i)|S_v(i) \tag{2.2}$$

Systolic Scan

Systolic Scan emulates the cardiac cycles of a beating heart. Figure 2.7 shows the design of systolic scan. The tiles are scanned from the inner layer to the outer layer, as denoted by different gray-levels in Figure 2.7. Without loss of generality, we describe the method with a simple case where $col_{max} = row_{max} = N$. Clearly, the length of scheduling bits in a scanning round is $\lceil N/2 \rceil$.

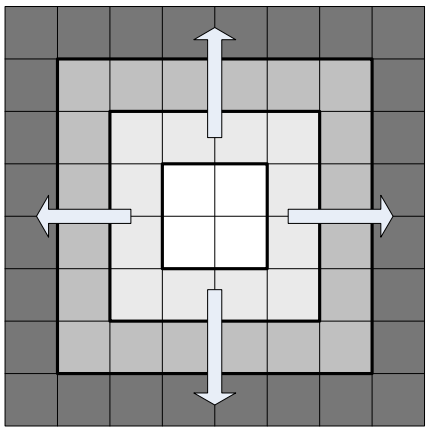


Figure 2.7: Systolic Scan

For the first time interval (represented by the first digit in the scheduling bits), the tiles at the center of the area set their first digit of scheduling bits to 1, and the scheduling bits for these tiles are $(\mathbf{1}\underbrace{000..000}_{\lceil N/2\rceil-1})^*$.

Similarly, for the n^{th} time interval, the tiles whose coordinates meet one of follow four conditions:

$$\begin{array}{llllll} row == n & \& & col > n-1 & \& & col \leq N-n \\ row == N-n & \& & col > n-1 & \& & col \leq N-n \\ col == n & \& & row > n-1 & \& & row \leq N-n \\ col == N-n & \& & row > n-1 & \& & row \leq N-n \end{array} \tag{2.3}$$

set their scheduling bits as follows:

$$\begin{aligned} S(i) &= (\underbrace{000..000}_{\lceil N/2\rceil-n-1}\ \mathbf{1}\underbrace{000..000}_{n})^* \\ i &= row * N + col \end{aligned} \tag{2.4}$$

where $n = 0, 1, 2..., \lceil N/2\rceil - 1$ and i is the index of tiles that satisfy the requirements.

Both line scan and systolic scan specify only the set of tiles need to be activated (covered) at a given point of time. The task of covering each tile set is accomplished by the second-level node scheduling, which is described in the next section.

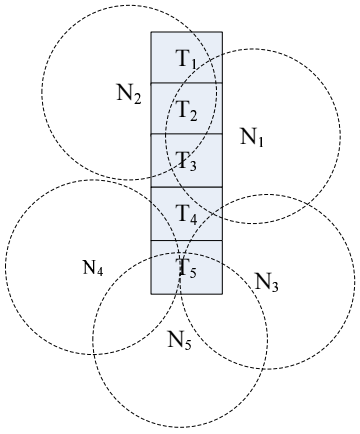


Figure 2.8: Physical Coverage

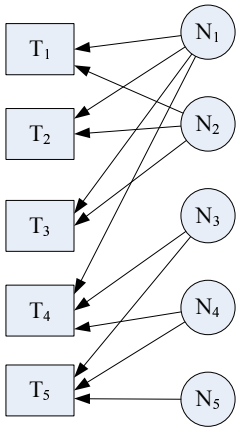


Figure 2.9: Bipartite Graph

2.3.3 Level II: Node Scheduling

Tile-level scheduling determines the set of active tiles TS_i at the time interval i. For example, in a horizontal line scan, the i^{th} column is activated at time interval i. In this section, we describe how we can translate a known tile schedule into a corresponding node schedule bits S, which can be interpreted directly by a generic switching algorithm.

Main Idea

Before we discuss the complete algorithm, we first illustrate our approach with a simple example. Figure 2.8 shows one column of tiles $TS = \{T_1, T_2, T_3, T_4, T_5\}$ that is covered by a set of nodes $NS = \{N_1, N_2, N_3, N_4, N_5\}$. Since we set the tile size smaller than the minimal target size, a tile is said to be covered as long as a portion of this tile is covered. Figure 2.8 can be mapped to the Coverage Bipartite Graph shown in Figure 2.9 according to the coverage relationship. Node scheduling consists of two steps. First, we keep identifying one-cover set with minimal number of nodes, until the size of one-cover set is above a certain threshold. For example, as shown in Figure 2.8, we identify three one-cover sets for TS: $CS^1 = \{N_1, N_5\}$, $CS^2 = \{N_2, N_3\}$ and $CS^3 = \{N_1, N_4\}$ to ensure that all nodes are used. Three sets CS^1, CS^2 and CS^3 can provide coverage to the tile set TS in a round-robin fashion. To do this, we create a node schedule that has three segments, each of which has a length of the tile schedule. If a node belongs to the CS^k set, the k^{th} segment has the same value as the tile schedule. Otherwise, the k^{th}

segment has an all-zero value. For example, if the tile schedule of TS is 0010, the final schedules for the nodes shown in Figure 2.8 are:

$$\begin{array}{lll} S_1 = (\underbrace{0010}_{1}\underbrace{0000}_{2}\underbrace{0010}_{3})^* & S_2 = (\underbrace{0000}_{1}\underbrace{0010}_{2}\underbrace{0000}_{3})^* & S_3 = (\underbrace{0000}_{1}\underbrace{0010}_{2}\underbrace{0000}_{3})^* \\ S_4 = (\underbrace{0000}_{1}\underbrace{0000}_{2}\underbrace{0010}_{3})^* & S_5 = (\underbrace{0010}_{1}\underbrace{0000}_{2}\underbrace{0000}_{3})^* & \end{array} \tag{2.5}$$

Identifying Minimum Set-Cover within Linear Time

To save energy at each time interval, we need to identify a minimum set of nodes to cover an active tile set. This is a typical set-cover problem, which can be formally defined as:

Definition 1 *Given a collection C of subsets of a finite set T, find a set cover C' ($C' \subseteq C$) for T, such that every element in T belongs to at least one member of C'.*

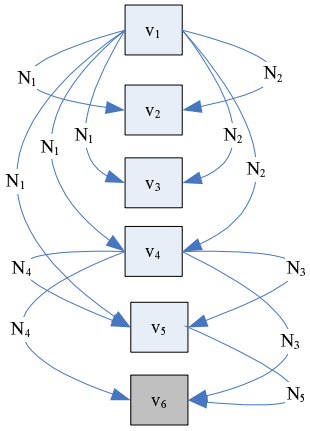


Figure 2.10: MSC using DAG

The generic Minimum Set Cover (MSC) problem has been proven NP-Hard and any polynomial algorithm can only find results of $1 + ln|T|$ optimum [25]. Fortunately, line scan coverage is a special case of the generic set cover problem, because a node can cover only a continuous segment of tiles. The main idea of our polynomial algorithm is to map Coverage Bipartite Graph (figure 2.9) into a Directed Acyclic Graph (DAG) (figure 2.10). The one-to-one mapping rules are as follows:

1. We map N tiles in TS_i into N vertices $V = \{v_1, ..., v_N\}$ and add one extra vertex v_{N+1}.

2. If a node covers a set of tiles $\{T_i, ..., T_{i+n}\}$, we create n directional edges (v_i, v_j) where $v_j = v_{i+1}, ..., v_{i+n+1}$. Each edge has a unit cost.

Through this mapping, the tile set cover problem can be reduced to the problem of finding out the shortest paths from v_1 to v_{N+1}. The mapping process takes $O(|V|) + O(|E|)$ time. For an arbitrary graph, the shortest path algorithm finishes within $O(|V|^2)$ using the Dijkstra algorithm. Since the graph we create is DAG, we can find the shortest path in $O(|E|)$ time, using a fast reaching algorithm [26].c Therefore the whole algorithm finishes in $O(|V|) + O(|E|)$.

To illustrate the idea, Figure 2.10 shows a DAG which is mapped from Figure 2.9. To identify the minimal set cover, we need to find out the shortest path from v_1 to v_6. In this simple case, we can cover all the tiles using one of following node sets: $\{N_1, N_3\}$, $\{N_1, N_4\}, \{N_1, N_5\}$, $\{N_2, N_3\}$ or $\{N_2, N_4\}$, which are five corresponding shortest paths from v_1 to v_6.

We note that the proposed polynomial algorithm does not apply to generic tile scheduling. When a tile set does not form a continuous curve or a node can cover multiple segments of a tile set simultaneously, the polynomial algorithm can not guarantee the complete coverage of active tiles. In these cases, we adopt a greedy set-cover method by choosing the node that covers the most number of tiles first.

Selecting Cover Sets for Multiple TS

Up to now, node scheduling has been described using a simple example that assumes a node only needs to cover one tile set. Obviously, to support line scan or systolic scan in a 2-D space, we need to identify cover sets for the whole area (not just for a single column). Thus a node may need to cover multiple tile sets TS_i. The detailed process to cover the area is as follows:

1. Each node maintains a counter SC to record how many times it has been selected into final Cover Sets (for the purpose of energy balance).

2. For a tile set TS_i, the algorithm calculates the minimum cover set MCS_i among the nodes with minimum SC values. If the nodes with minimum SC values can not form a complete cover set, nodes with higher SC values are used.

3. After we obtain all MCS_i, the smallest eligible MCS_i (SMCS) is selected and recorded for purpose of node scheduling, and the SC values of nodes within this SMCS set are incremented.

4. Each TS_i has a coverage threshold, denoting the maximum number of nodes that can be used in a selected MCS_i. These thresholds are calculated based on the concept of redundancy. If the first MCS_i chosen for TS_i includes M nodes, the number of nodes in the following MCS_i for TS_i should not differ significantly, in order to reduce redundancy in coverage. We set the threshold for TS_i as $M \times \frac{2\pi}{\sqrt{27}}$, according to the redundancy in circle covering [27].

5. The SMCS selection process is repeated until the size of all MCS_i are larger than their thresholds.

Create Node Scheduling Bits R

Suppose K one-cover sets are selected for a tile set TS_i with a tile schedule S_T (calculated in Section 2.3.2), we create a node schedule S_i for node N_i, which has K segments. The value of each segment is either S_T or zero. If a node belongs to the k^{th} one-cover set, the value of the k^{th} segments is S_T. Otherwise, the k^{th} segment has an all-zero value. Since in a 2-D space, a node might need to cover different tile sets in a single round. Supposing a node needs to cover M different tile sets, the final node schedule S is:

$$S = \overset{M}{\underset{i=1}{OR}}(S_i)^*. \tag{2.6}$$

The complete operation of the node-level scheduling is shown in Algorithm 2.3.3.

2.3.4 Optimizations and Extensions

For the sake of clarity, we described the basic design without optimization and extensions. In this section we discuss a set of optional enhancements to improve the basic design.

Algorithm 1 uSense Global Scheduling

Input: TS_i:The set of active tiles at time i
Input: NS_i:The set of nodes that can cover the tiles $\in TS_i$
where $i = 0, 1, ..., n-1$
Output: Node Schedule R

```
Set the counter SC of all nodes zero
Find the minimum cover set MCS_i for individual TS_i
Threshold of TS_i = |MCS_i| × 2π/√27
repeat
    Index = 0; Size_min = ∞
    for i = 0 to n − 1 do
        Find the new Minimum Cover Set MCS_i for TS_i from the nodes with minimum
        SC values
        if (|MCS_i| < Size_min) and (|MCS_i| < Threshold of TS_i) then
            Index = i; Size_min = |MCS|
        end if
    end for
    Increment the SC values of the nodes ∈ MCS_Index
    CS_Index = CS_Index ⋃ {MCS_Index}
until (∀TS_i, |MCS_i| ≥ Threshold of TS_i) OR (Size_min = ∞)
Convert Tile Schedule into Node Schedule using CS_i
```

Known Mobility Pattern

To support all possible cases, uScan assumes the mobility pattern of the targets is unknown. Evidently, we can achieve a better performance if such information is available. Here we illustrate the idea with line scan.

- **When the Target Direction is Known**, line scan can choose the scan direction accordingly. For example, if the targets only enter from the left edge and exit from the right edge, we should scan from right to left. There are two operation modes for line scans to guarantee detection. In the first mode, the switching rate of line scan is very low. It turns on the leftmost column of tiles first. The i^{th} column is turned on only when the $(i-1)^{th}$ column runs out of energy. The benefit of this approach is less energy consumption for state transitions, however the detection delay degrades overtime and it cannot detect static events quickly. The second mode is the normal scan at a fast switching rate. With the target speed r tiles per second and switching rate R tiles per second, in a network of N by N tiles,

the average detection delay is $\frac{N}{r+R}$. If direction information is unknown, the scan direction could be wrong, which leads to a higher delay such as $\frac{N}{R-r}$ or even no detection when $r > \frac{R}{2}$ if the direction of scan is the same as the target direction.

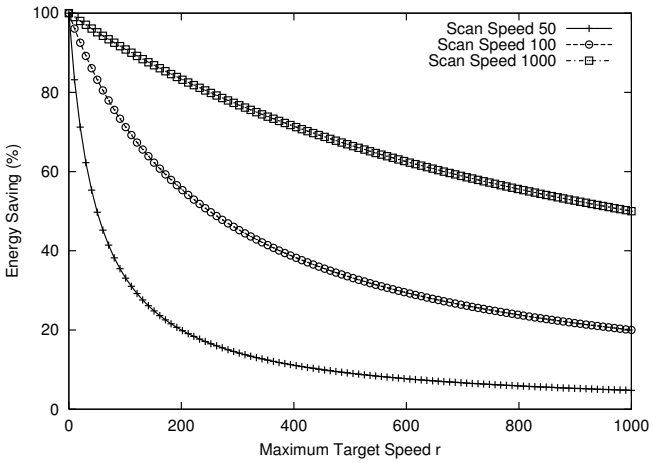


Figure 2.11: Energy Saving under different target speeds

- **When both the target direction and maximum target speed are known,** line scan can save more energy. Suppose a target enters from the left edge and exists from the right edge with maximum target speed r tiles per second. Given a network of size of N by N tiles, it takes at least $\frac{N}{r}$ seconds for a target to move from left to right. In the basic design, line scan is required to turn on one column at any given time. However with more information, this requirement becomes unnecessary. Between consecutive rounds of scans, we can allow a time gap of $\frac{N}{r}$, during which whole network is turned off. Since it takes $\frac{N}{R}$ seconds to finish a round, we save $\frac{R}{R+r}$ percentage of energy compared with the basic design. Figure 2.11 shows the percentage of energy savings under different target speeds and scanning speeds. One key observation is that when target speed is very slow, the energy saving approaches 100%. Even as the target speed equals the scanning speed, 50% energy can be saved.

Reducing 0/1 and 1/0 Transitions

In the basic design, we ignored the energy consumption during the 0/1 or 1/0 state transitions. This is a reasonable approach because we found that the transient energy consumption in embedded devices such as mica2 and XSMs is very small [anonymous]. However, if the switching rate is very high, it is not negligible. Some techniques can be applied to reduce the state transitions. A straightforward solution is to reduce the switching rate, especially when the target speed is slow and the detection delay is less critical compared with saving energy. The second approach is to assign consecutive active intervals to a node. For example, a (01100)* schedule introduces half the number of transitions as a (01010)* schedule. To achieve this, we adopt a post-optimization method. Essentially, we exchange the active schedule bits among pairs of nodes that cover the same set of tiles. This type of exchange does not change how tiles are covered and also does not change the duty cycle of individual nodes. For example, if two nodes cover the same set of tiles and have the schedules (01010)* and (10101)*, respectively, we can exchange the schedules into (10001)* and (01110)* to reduce the number of state transitions.

Support Differentiated/Robust Surveillance

Differentiated surveillance [8] can be supported easily by uScan, due to its set-cover based approach. Instead of turning on one set of nodes to cover a column/row, uScan can turn on multiple disjoint set of nodes to increase the degree of coverage. This leads to a higher detection confidence, but at the cost of network lifetime. Similarly, fault tolerance can be achieved by turning multiple sets on. It is interesting to emphasize that nodes actually have no concept of set, which leads to a nice property for fault tolerance: To fix the failure of nodes, we only need to modify the scheduling bits S of the nodes in the neighborhood of failed node and no coordination between nodes is needed.

Considering the Local Coverage Density

In the basic design, we balance the duty cycle among the neighboring nodes, using the Set Counter (SC). This method works well when nodes are roughly uniformly distributed. When network deployment is highly irregular (even in the neighborhood), it

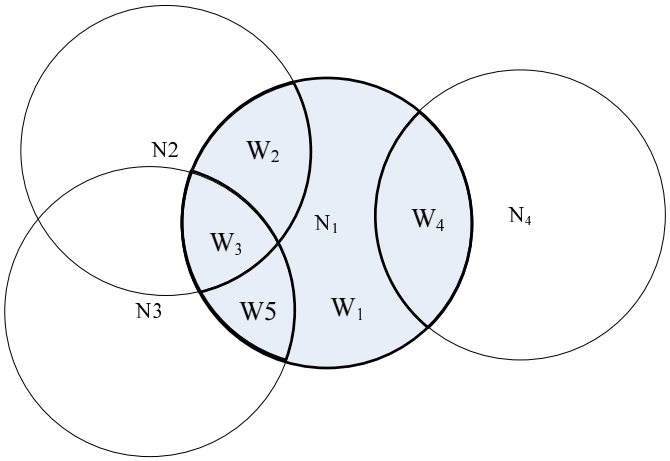


Figure 2.12: Calculate Local Density

is beneficial to take local coverage density into account. Local Coverage Density (LCD) is defined as the weighted average degree of coverage within a node's sensing range. Figure 2.12 shows how LCD of Node N_1 is calculated. The boundaries of four nodes partition the sensing area of N_1 into 5 sub-areas with size of W_1, W_2, W_3, W_4 and W_5. In this scenario, the LCD of $N_1$is:

$$LCD = \frac{W_1 + 2(W_2 + W_4 + W_5) + 3W_3}{W_1 + W_2 + W_3 + W_4 + W_5} \tag{2.7}$$

The main idea is to select the node with the highest LCD first, if all candidate nodes have the same Set Counter (SC) value. By doing so, the nodes in a denser area have a higher chance to be selected than the nodes in the sparser area.

2.4 Design Analysis

Different from full coverage algorithms in [7, 28], uScan covers only a part of a network. On one hand, this approach significantly increases the network lifetime, but on the other hand, it introduces a certain delay in target detection. In this section, we provide analytic results on the performance of uScan. Here, we focus on the tile-level analysis instead of the node-level. Let's consider an area with N by N tiles.

2.4.1 Network Lifetime

In line and systolic scan methods, a tile is turned on once per round. Supposing a tile can be turned on at most K time, the network lifetime is the product of K and the delay in each scanning round. The delay of each round is $\frac{N}{R}$, where R is the switching rate. Therefore the network lifetime is $\frac{NK}{R}$ under line scan and $\frac{\lceil N/2 \rceil K}{R}$ under systolic scan. Under the same configuration, the network life time for full coverage algorithms is approximately $\frac{K}{R}$. The network lifetime ratio between uSense and full coverage is $O(N)$, which indicates that the benefit of uScan increases with the network size.

2.4.2 Detection Delay for Static Targets

To evaluate the detection delay for static targets, we assume that a target is randomly located in an area and is detected after a neighboring node turns on for $\frac{1}{R}$ seconds. In line and systolic scan, in order to guarantee detection, a tile must be turned on once per round. The minimal detection delay happens when a target shows up in a tile right before this tile is turned on. In this case, the detection delay is $\frac{1}{R}$. The maximum detection delay happens when a target shows up in a tile right after this tile is turned on. In this case, the detection delay is $\frac{1+N}{R}$ for line scan and $\frac{1+\lceil N/2 \rceil}{R}$ for systolic scan. Since the delay is uniformly distributed in a round, the expected delay is $\frac{2+N}{2R}$ for line scan and $\frac{2+\lceil N/2 \rceil}{2R}$ for systolic scan. Under the same configuration, the detection delay for full coverage algorithms is zero. To reduce the detection delay in uScan, we can divide a network into sub-networks, where multiple line scans and systolic scans are executed in parallel.

2.4.3 Breached Area for Mobile Targets

In a full coverage scenario, the worst-case breach area is zero. A mobile target is detected once it enters into the area. In the scanning approach, a target would reach a certain portion of the area before it is detected. We define the largest percentage of the area that a target can reach without being detected as the Worst-Case Breach (WCB). A smaller WCB indicates a better performance in mobile target detection. To calculate WCB, we assume the following mobility model: A target can only enter from outside of the network, and the maximum speed of any target is r tiles per second.

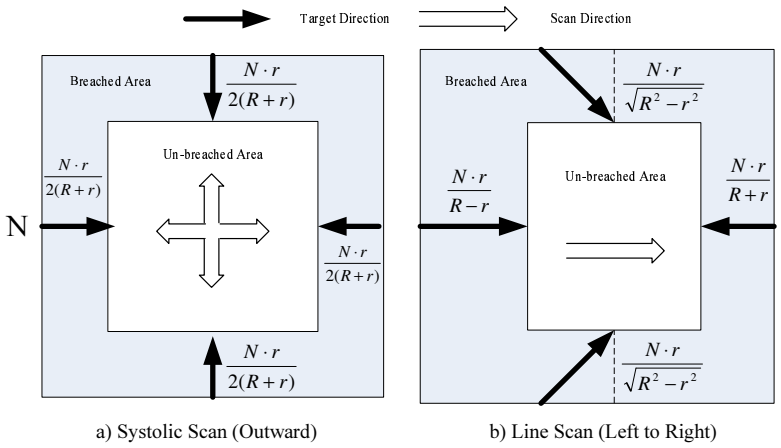


Figure 2.13: The Breached Area

The worst case breach scenario for systolic scan (outward direction) is shown in Figure 2.13(a). In this scenario, the worst-case breach happens when a target enters at the beginning of the scan round. The distance this target can breach without detection is $\frac{Nr}{2(R+r)}$. Therefore, the $WCB_s(r, R)$ for systolic scan under the target speed r and switching R is:

$$WCB_s(r, R) = \frac{(2R + r)r}{(R + r)^2}. \tag{2.8}$$

The worst case breach scenario for line scan is shown in Figure 2.13(b), which has four more sophisticated cases:

1. If a target enters from the left edge (the same direction as the direction of scan), the distance this target can breach is $\frac{Nr}{R-r}$.

2. If a target enters from the right edge (opposite direction as the direction of scan), the distance this target can breach is $\frac{Nr}{R+r}$.

3. If a target enters from the top or bottom edge. In order to achieve the maximum breach, a target should enter with an angle $\alpha = \arctan(\frac{\sqrt{R^2-r^2}}{r})$. In this case, the maximum distance that this target can breach is $\frac{Nr}{\sqrt{R^2-r^2}}$.

4. If a target has a speed of $(\sqrt{2}-1)R$ or greater, it can enter the area from the left to reach at least $\frac{1}{\sqrt{2}}$ percent of the area and it can also enter the area from the

right to reach at least $1 - \frac{1}{\sqrt{2}}$ percent of the area. Therefore, the whole area is breached.

By combining these four cases, we get $WCB_l(r, R)$ for line scan , assuming target speed r and switching R:

$$WCB = \begin{cases} \frac{2Rr}{R^2-r^2} + \frac{2r}{\sqrt{R^2-r^2}}(1 - \frac{2Rr}{R^2-r^2}) & r < (\sqrt{2}-1)R \\ 1 & r \geq (\sqrt{2}-1)R \end{cases} \tag{2.9}$$

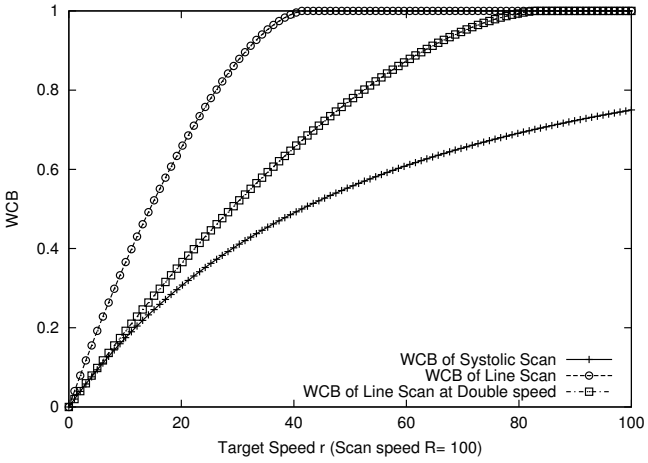


Figure 2.14: Performance Comparison

Now we are ready to compare two global scheduling algorithms. As shown in Section 2.4.1, for a given switching rate R, systolic scan consumes twice as much energy as line scan does. To obtain a fair comparison, we thus double the switching rate of line scan. By comparing WCB_l and WCB_s, it is easy to prove that $WCB_l(r, 2R) \geq WCB_s(r, R)$ at all target speeds. In other words, systolic scan is better than line scan in terms of minimizing the breaching area. Actually we have proven that systolic scan is an optimal scanning algorithm in terms of preventing area breach when the target speed r is very fast (see appendix A). On the other hand, the line scan algorithm has its own advantages. As we have shown in Section 2.3.3, we are able to obtain optimal set-cover results for line scan within a polynomial time. To illustrate the difference further, Figure 2.14 shows the WCB values under different target speeds r (0-100), when the switching rate R is 100. Clearly, the difference is significant. For

example, when the target speed is half of scanning speed(r=50), systolic scan protects about half of the area, while line scan cannot protect any portion of the network.

2.5 Implementation and Evaluation

We have implemented a complete version of uSense (with uScan) as designed in Section 2.2 and 2.3.

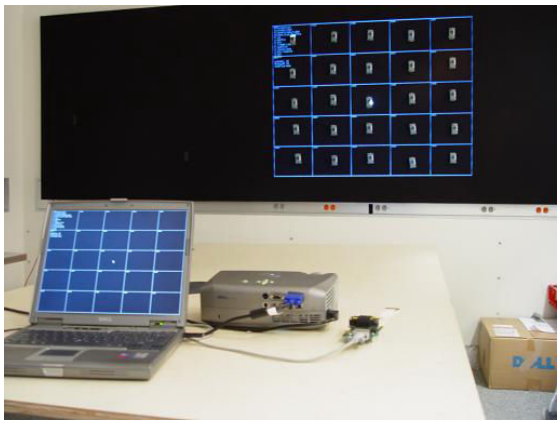


Figure 2.15: uSense System Setup

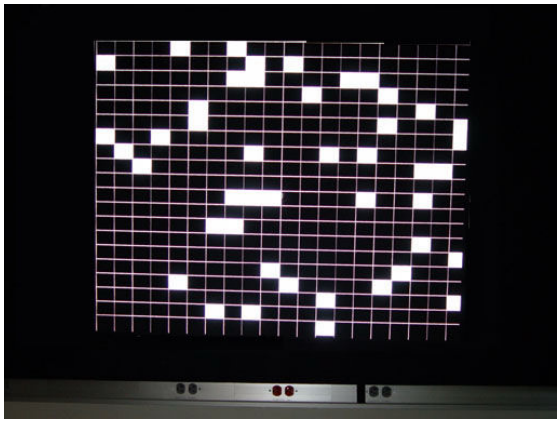


Figure 2.16: Injecting Events Display

- **The generic switching algorithm** is written with the NesC language, running on the TinyOS/Mote platform. The compiled image of a full implementation occupies 21,040 bytes of code memory and 907 bytes of data memory. A simple timer-driven FA logic is implemented to turn a mote on/off according to the

scheduling bits. We use FTSP [29] for the purpose of time synchronization among motes and Deluge [20] for the purpose of wireless reprogramming. The synchronization accuracy is at tens of microseconds, which is sufficient for most sensing scheduling algorithms.

- **The scheduling algorithms** are written in Java, runs on a laptop. Since sensor nodes have no concept of scheduling, the global scheduling algorithm uScan and other coverage algorithms are written only in Java. The scheduling bits S and the switching rate R are disseminated from the base, using a single packet. We also implement an evaluation engine using Java, which generates virtual targets using the light. To accurately measure the delay, we implement an NTP-like two-way handshaking synchronization protocol over a serial cable to synchronize the base mote and laptop. This synchronization protocol is not part of uSense and is only used for the evaluation purpose.

As shown in Figure 2.15 to evaluate uSense and uScan, we attach 25 MicaZ motes on a veltex board (4 feet by 12 feet) using velcro straps. We use a DELL 2300MP projector to generate light spots on the veltex board. These light spots are used to emulate static and mobile events. For example, the static events are randomly generated in the grid to trigger the detection, as shown in Figure 2.16. After the nodes detect the light events, they report timestamps to the laptop, where the delay is calculated.

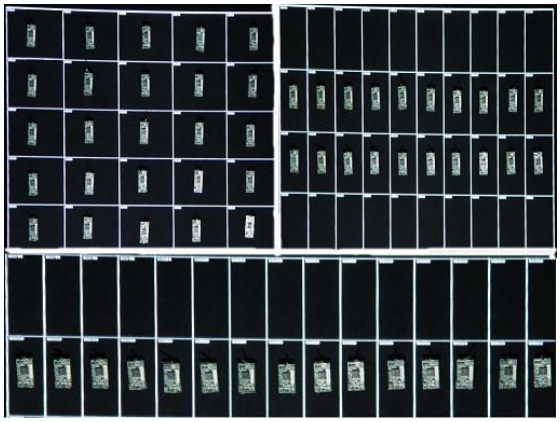


Figure 2.17: 5×5, 2×10, 1×15 Layouts

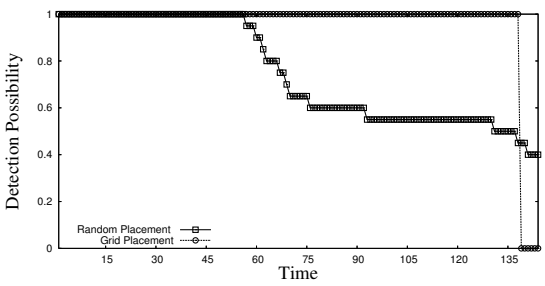


Figure 2.18: Detection Possibility Over Time under Different Node Placements

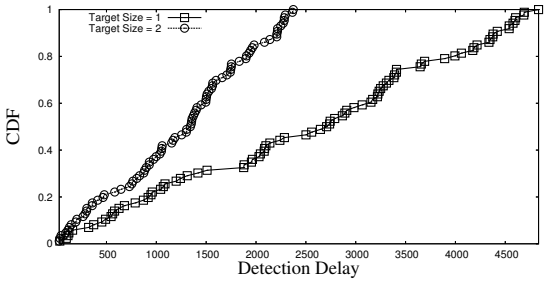


Figure 2.19: Detection Delay of for Static Targets under Different Target Sizes

As shown in Figure 2.17, three grid layouts are used in the experiments: 15 by 1, 10 by 2 and 5 by 5. In addition, we evaluate uScan with random placement as well. The locations of nodes in the random placement are obtained through a random generator. Each node is assigned an energy budget (the number of times a node can be turned on), which is used to evaluate the lifetime of the sensors. Events are repeated hundreds of times to obtain results with high statistical confidence.

2.5.1 Testbed Evaluation

The system evaluation focuses on the detection delay for static and mobile targets, the detection probability for mobile targets and the node lifetime. All the experiment are repeated 10 times.

Detection Probability Over Time

In this experiment, we inject static targets through the DELL projector into the network to evaluate the detection probability over time. A target is missed only if a tile is not covered (i.e., a node runs out of power). We test uScan under the random placement and grid placement using 10 MicaZ motes. The results are shown in Figure 2.18. Since nodes in the grid placement have well balanced duty-cycles, they provide full coverage until all of them run out of energy simultaneously. In the random placement, the sparser areas become uncovered first, and coverage degrades gradually over time. For example, random placement still keeps about 40% coverage when coverage reduces to zero in the grid placement.

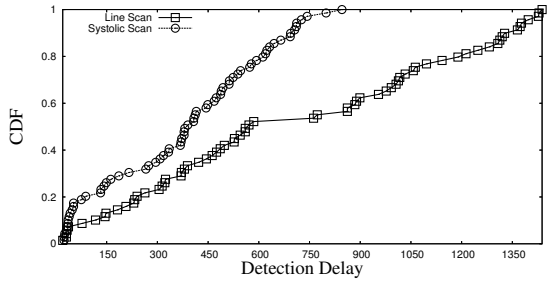


Figure 2.20: Detection Delay of Line and Systolic Scan for Static Targets

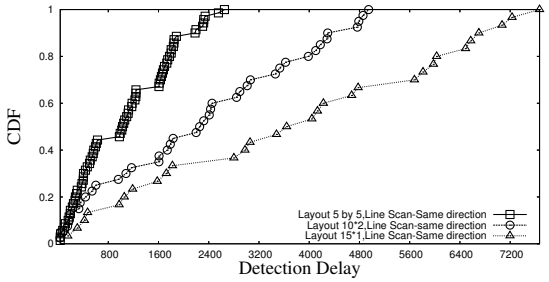


Figure 2.21: Detection Delay of Line Scan for Mobile Targets

Detection Delay for Static Targets

In this experiment, we investigate the detection delay for static targets under different minimum target sizes. Static targets are injected at random time intervals into the area. Since the tile size is determined by the minimum target size, the number of columns needed to cover the same area reduces when the minimum target size increases, therefore the detection delay reduces as well. Figure 2.19 shows the Cumulative Density Function (CDF) curves of the delays for two target sizes. Clearly, a larger target size leads to smaller delays. For example, when the target size is one, the maximum delay on detection is $4834ms$, while the maximum delay is $2378ms$ with a target size of two.

Comparison of Detection Delay

In this experiment, we use 25 MicaZ to form a 5 by 5 grid and compare the detection delay of static targets for line and systolic scan, again using CDF curves. From Figure 2.20, we can see that under the same switching rate, the detection delay of systolic scan is about one-half of line scan, which is consistent with the length of scheduling bits shown in Section 2.3.2. In the 5 by 5 grid layout, the average detection delays for systolic and line scan are $380ms$ and $706ms$, respectively.

Impact of the Network Size and Scan Direction

In this experiment, we study the impact of the network size using three network layouts. The target moves in the same direction as line scan. As shown in Figure 2.21, as the network size reduces, the detection delay decreases accordingly. For example, the

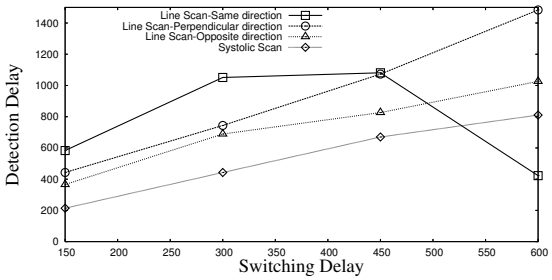


Figure 2.22: Detection Delay Under Different Switching Delays

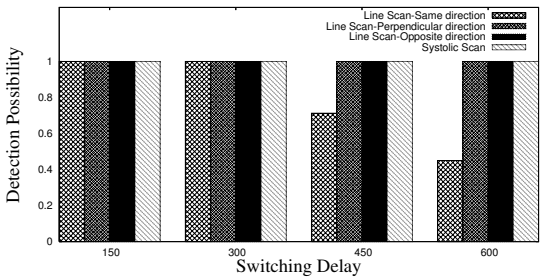


Figure 2.23: Detection Possibility Under Different Switching Delays

average detection delays for the 1 by 15, 2 by 10 and 5 by 5 layouts are $3758ms$, $2284ms$ and $1051ms$, respectively. This indicates that to guarantee a certain detection delay, we should partition a large area and perform scans within the sub-areas.

Impact of the Switching Delay and Scan Direction

Figure 2.22 studies the detection delay under different switching delays (the reciprocal of the switching rate R) and different target directions. This investigation is intended to reveal the importance of designing fast hardware and detection algorithms. We use a 5 by 5 layout, generate mobile targets from three different directions, and measure the delays before the mobile targets are detected. Figure 2.22 shows four curves, representing three target moving directions on line scan and one direction on systolic scan. Systolic scan has the smallest detection delay at all switching rates. Line scan in the opposite direction of a target moving direction provides the second smallest detection delay. The longest delay happens when we scan at the same direction as the target moving direction.

In addition, Figure 2.22 shows that, generally, when the switching delay increases, the detection delay increases linearly. Interestingly, there are two data points that do not follow this trend. It is because that line scan misses the targets when the scan speed is below twice the target moving speed (when both move in the same direction). In our setup, this happens when the switching delay is longer than 300ms. Under these slow scanning speeds, uScan may miss targets and record only the short detection delays, leading to a small average detection delay. This is also confirmed by the detection

probability results in Figure 2.23, which indicate when targets move oppositely to the direction of line scan, we can ensure the 100% detection of mobile targets. However, if the scanning direction is the same as the target moving direction, the detection probability drops to 45% at the long switching delay of $600ms$, as shown in Figure 2.23.

2.6 Large Scale Simulation: Comparing with State-of-the-Art

Experiments on the test-bed indicate that uSense can be efficiently implemented on the resource constrained devices and reveal its nice features. In this section, we compare the performance of uScan with several state-of-the-art sensing coverage algorithms integrated into uSense architecture, validate the benefit of asymmetric sensing architecture and the two-level scheduling approach.

In this simulation, up to 10,000 sensor nodes are randomly distributed in a 300m×300m square field. The sensing range is 10m. The sensor nodes are deployed with a random distribution into the square field. To avoid performance distortion due to the edge effect, we set the coverage area as the 290m×290m square in the center of the square field and do statistics on the central 275m×275m field. The following baselines and bounds are adopted for purposes of comparison:

1. **Baseline-I: All-Working:** The full coverage mode with all nodes on.
2. **Baseline-II: DiffSurv:** Differentiated Surveillance for sensor networks proposed in [8] integrated into the uSense architecture.
3. **Baseline-III: Virtual Patrol:** Coverage-oriented patrols using wireless sensor networks proposed in [30] integrated into the uSense architecture.
4. **Upper Bound-I:** Optimal full coverage using $2/\sqrt{27}r^2$ circles with a honeycomb layout.
5. **Upper Bound-II:** Ideal upper bound for line scan, which assumes that nodes are placed optimally such that $|MCS_i|$ for each time interval is minimized.

All the experiments are repeated 100 times with different random seeds and different node deployments. The following performance metrics are used to evaluated the performance of uScan.

1. **Network Half-life:** We define the half-life of a sensor network as the time from the beginning of the deployment until exactly half of the sensor nodes are still alive. This metrics indicates the energy efficiency of the network as a whole.

2. **Coverage Overtime:** This metrics indicates the coverage percentage over time (e.g. full coverage equals 100%)

2.6.1 Performance under Full Coverage Mode

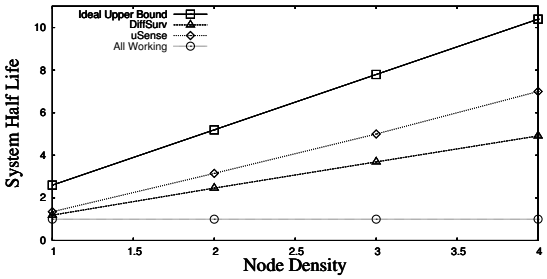


Figure 2.24: System Half Life vs. Node Densities

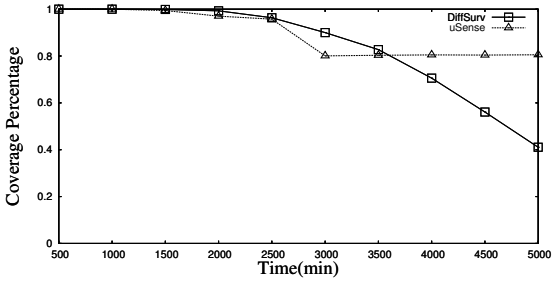


Figure 2.25: Sensing Coverage over Time (Density = 3)

If the tile set TS to be covered includes all the tiles in the network, uScan essentially provides a full coverage. In this experiment, we compare uScan with other solutions under Full Coverage Mode to evaluate the effectiveness of the set-cover approach in uScan. Figure 2.24 shows the network half life of four different solutions: uSense, DiffSurv, all-working lower bound and theoretical upper bound (by assuming a honeycomb layout). From Figure 2.24, we can see that the half lives for all cases are increasing linearly when the node density increases. The slope of uSense is larger than DiffSurv, which implies that as the node density increases, the difference in half-life between uSense and DiffSurv increases as well. For example, at the node density 1, the half life for DiffSurv and uSense are 1.19 and 1.35, respectively. And when the node density reaches 4, the corresponding two half lives are 4.91 and 6.99 and the half-life difference increases from 13.4% to 42%.

Figure 2.25 shows the sensing coverage of uSense and DiffSurv over time. Here we assume if a node is turned on all the time, it can last for 1 unit of time (1000 minutes). From Figure 2.25, we can see that uSense can sustain for a longer time than the DiffSurv. The coverage of DiffSurv drops considerably after 3500 minutes. On the contrary, the coverage of uSense stay stable after 3500 minutes. The reason for uSense has less coverage than DiffSurv between time 2500 and 3500 is that each cover set for uSense has less redundancy than DiffSurv, thus the effect of dead nodes has relatively larger impact on the coverage. However, uSense significantly improved the average life for single node and the half life for the whole network, which leads to a steady staircase curve compared to the continuous fast drop in DiffSurv.

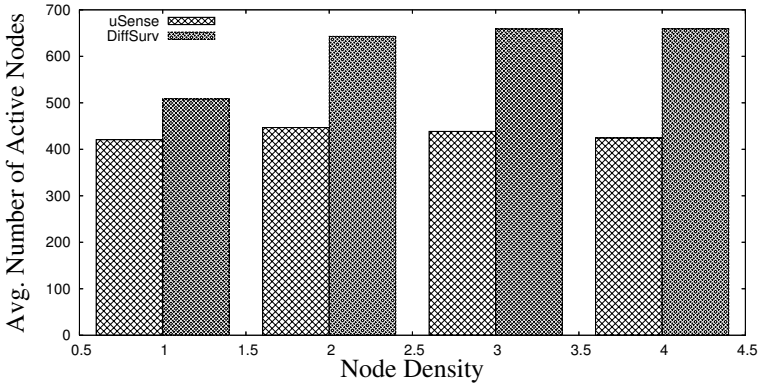


Figure 2.26: Number of Avg. Active Nodes vs. Node Densities

Figure 2.26 studies the average number of active nodes at any time instance under different node densities. At all node densities, the DiffSurv activates more nodes than the uSense. As the node density increases, the average number of active nodes for DiffSurv increases slightly, while uSense has decreased number of active nodes. The reason for uSense can decrease the number of active nodes as node density increases is the increasing number of nodes deployed in the field offers the node scheduling algorithm proposed in Section 2.3.3 better opportunities to generate more energy efficient cover sets. In contrast, the integration of working schedules at each individual node for DiffSurv may offset some benefits gained with increasing node density.

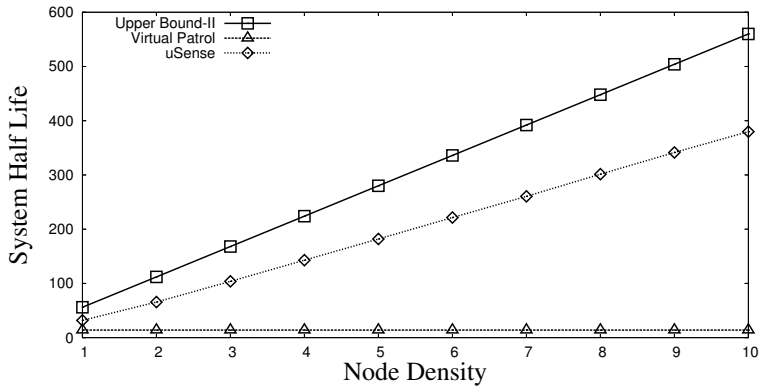


Figure 2.27: System Half Life vs. Node Densities

2.6.2 Performance under Scanning Mode

In [30], Gui and Mohapatra propose a virtual patrol solution similar to our line scan scheme. The main difference between virtual patrol and uScan is that virtual patrol only provides single-level of scheduling. At each time, if a node's distance to the patroller is within its sensing range, the node is active; otherwise it is in sleep state. Figure 2.27 shows the half life of the line scan for uSense, virtual patrol and ideal upper bound. From the Figure we can see that as the node density increases, the system half life of the uSense increases almost linearly. On the contrary, the half life of the virtual patrol remains steady and cannot take the advantage of the increased node density. When the node density reaches 10, the half life of the uSense is 379.88, while the virtual patrol has a half life of 13.99, which is about 27 times energy inefficiency than the uSense. The major reason for such large performance gap is that virtual patrol activates all the nodes can sense the patroller, while the uSense only use a minimized subset of the eligible nodes to provide the desired coverage.

2.7 Large Scale Simulation: Investigating Protocol Performance and Sensitivity

Section 2.6 show the combination of uSense and uScan outperforms the existing state-of-the-art sensing coverage protocols both in a full coverage case(DiffSurv [8]) and a partial coverage case (virtual patrol [30]). In this section, we intend to investigate how

sensitive and how robust our designs are, in order to assist the users of uSense to make guided decisions on system configurations.

2.7.1 Simulation Setup and Baselines

The simulation setup in this section follows the description in Section 2.6 with 10,000 nodes. All the experiments are also repeated 100 times with different random seeds and different node deployments. The 95% confidence intervals are within 1∼15% of the mean, which is not plotted for the sake of legibility. The following performance metrics are used to evaluate the performance.

1. **Network Half-life:** This metrics indicates the energy efficiency of the network as a whole.
2. **Node Energy Consumption:** This metrics indicates the energy efficiency of individual nodes.
3. **Coverage Overtime:** This metrics indicates the coverage percentage over time (e.g. full coverage equals 100%)
4. **Detection Delay for Static Targets:** This metrics indicates the detection performance in static scenarios.
5. **Worst-Case Breach (WCB) for Mobile Targets:** This metrics indicates the detection performance in dynamic scenarios.

2.7.2 Protocol Performance

In this section, we compare the performance of uScan with the lower-bound, upper-bound and several versions of uScan to investigate its performance under different system configurations.

Impact of the Node Density

Figure 2.28 shows the impact of the node density on the lifetime of nodes for line scan scheme. From the figure, we can see that similar to the whole coverage discussed in Section 2.6, the lifetime of a single node for uSense increases linearly as the node

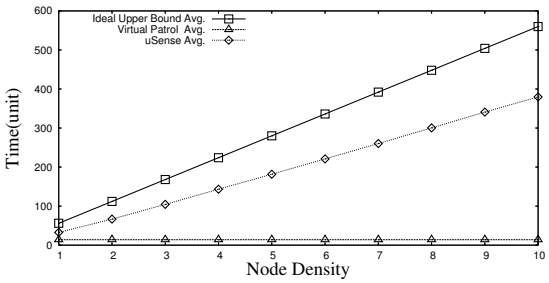


Figure 2.28: Avg. Single Node Life vs. Node Densities

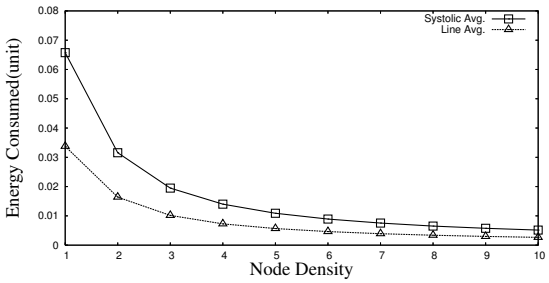


Figure 2.29: Avg. Energy Consumption vs. Node Densities

density increases. In contrast, the virtual patrol scheme, which activates all nodes to participate in the coverage, cannot take advantage of the increasing node density. We note the upper-bound can only be obtained by assuming a perfect honeycomb layout, which is a loose bound impossible to achieve with a random layout. Our experiments also reveal that when nodes in uSense are placed with a honeycomb layout, uSense achieves identical performance as the upperbound.

Figure 2.29 compares the single node energy consumption for two scanning schemes, line scan and systolic scan. The single node energy consumption for both line and systolic scan decreases as the node density increases. The energy consumption for systolic scan is about twice that of line scan, which is consistent with the analytical results in Section 2.4.

Impact of Minimum Target Size

In uScan, the tile size is set to the minimum target size in order to guarantee the detection. Figure 2.30 studies the impact of the minimum target size on single node life for line scan. From the figure, we can see that node life generally increases as the target size becomes larger in the ideal upper bound and uSense. However since virtual patrol doesn't support rotation, the node life time does not change. For example, even at minimum target size 1, the average single node lifetime for uSense and virtual patrol are 105 and 14, respectively, about a 7.5 times difference.

Figure 2.31 compares single node energy consumption in line and systolic scan. For both line and systolic scan, the single node energy consumption drops as the minimum target size increases. However, at each minimum target size, systolic scan consumes

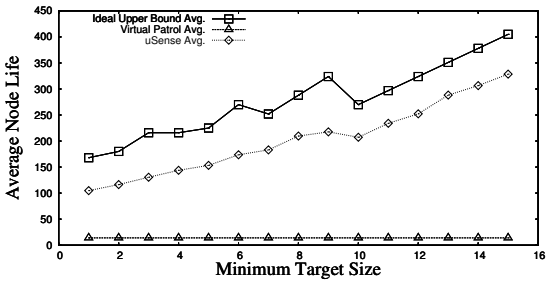


Figure 2.30: Avg. Node Life vs Target Size (Density = 3)

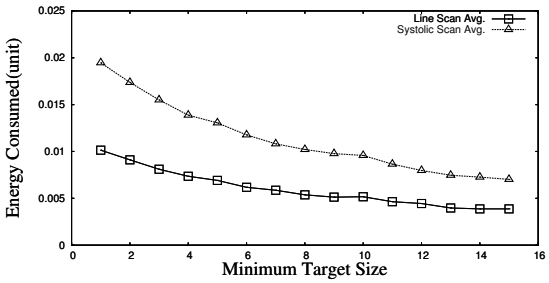


Figure 2.31: Avg. Node Energy Consumption (Density = 3)

about twice energy as line scan. For example, at a minimum target size of 8, the energy consumption for line and systolic scan are 0.0018 and 0.0033, respectively.

Detection Delay for Static Targets

In this experiment, we study the coverage performance of static target detection. Figure 2.32 shows the detection delay for line and systolic scan under different minimum target sizes. From this figure, we can see that the detection delays for both line and systolic scan decrease as the minimum target size increases. And for the same minimum target size and switching rate, systolic scan can detect targets about twice as fast as line scan. At a minimum target size of 8, the average detection delay for systolic scan is 6.46, while the delay for line scan is 15.10.

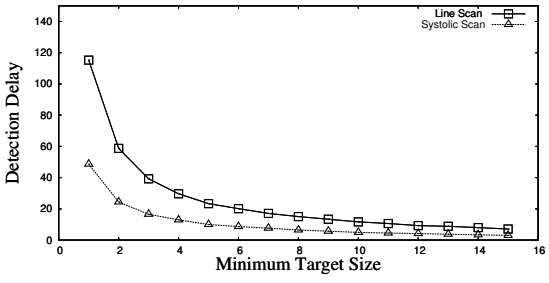


Figure 2.32: Avg. Static Targets Detection Delay (Density = 3)

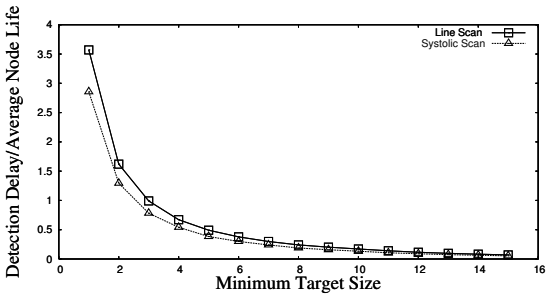


Figure 2.33: Delay and Lifetime Ratio for Static Targets

In Figure 2.33, we study the energy efficiency for line and systolic scan by investigating the ratio of the detection delay and node lifetime. From the figure, we can see that

for every unit of lifetime, line scan yields a longer detection delay than systolic scan, and this implies that systolic scan is more energy efficient than line scan in reducing the detection delay. This result actually contradicts the results we obtained from analysis. The main reason is that when the minimum target size is small, it is possible that a node in systolic scan covers multiple tile segments, which we didn't take into account in the analysis. However, when the minimum target size increases, such probability reduces. Therefore, line and systolic scan achieve same efficiency as shown in Figure 2.33. This trend is nicely predicted by our analysis.

Worst Case Breach for Mobile Target

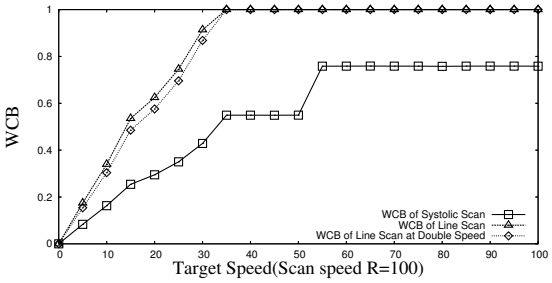


Figure 2.34: Breach Percentage vs. Target Speed

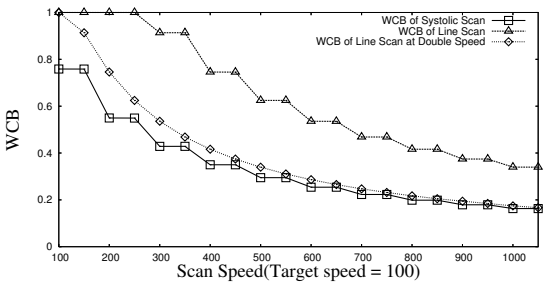


Figure 2.35: Breach Percentage vs. Switch Rate

This experiment is used to evaluate how our analytical results reflect reality. Figure 2.34 exhibits the same pattern as the Figure 2.14 in Section 2.4. One interesting difference is that Figure 2.34 presents a zigzag curve instead of a smooth curve as shown in Figure 2.14. This is expected because in analysis, we assume each tile is very small (i.e., $N \to \infty$). Figure 2.34 confirms that systolic scan provides a better protection than line scan. Figure 2.35 indicates that the scanning speed always helps to reduce the WCB. Therefore, it is very beneficial to design fast hardware and detection algorithms.

Coverage Over Time

Coverage over time is an important metric to evaluate the system performance during its lifetime. Figures 2.36 and 2.37 investigate coverage over time for line and systolic scan under different node densities. For both line and systolic scan, the higher node

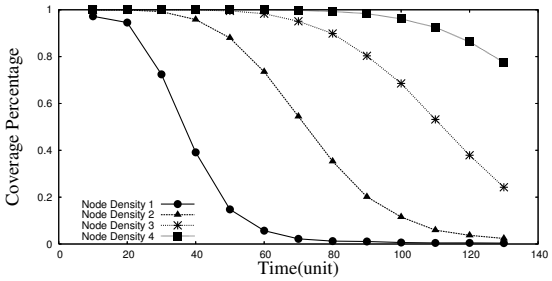


Figure 2.36: Line Scan Coverage over Time (Target Size=1)

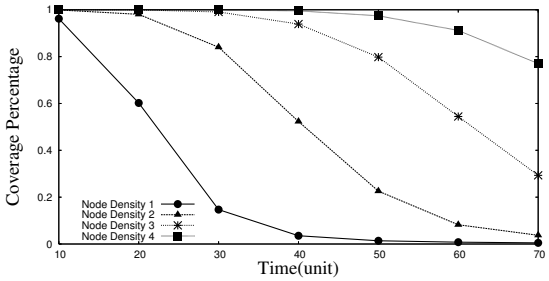


Figure 2.37: Systolic Scan Coverage over Time (Target Size=1)

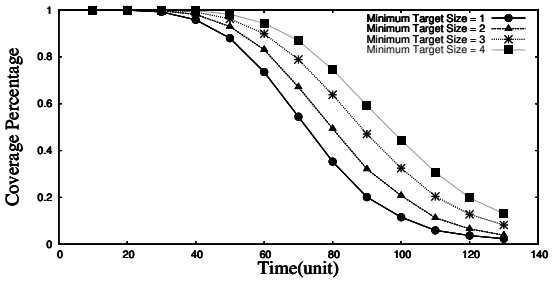


Figure 2.38: Line Scan Coverage over Time (Density = 2)

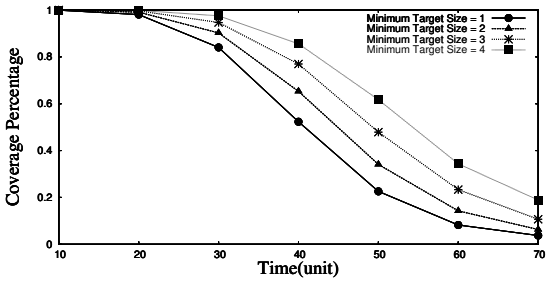


Figure 2.39: Systolic Scan Coverage over Time (Density = 2)

density leads to the better coverage over time. And since the energy consumption for systolic scan is about twice that of line scan, the system using line scan can sustain longer than the one using systolic scan.

Figures 2.38 and 2.39 show the system coverage over time for line and systolic scan under different minimum target sizes. From the figures, we can see that as the minimum target size increases, we obtain better coverage over time for both line and systolic scan.

2.7.3 Protocol Sensitivity

In this section, we study the sensitivity of uSense in the presence of two reality issues: the precision of network synchronization among deployed sensor nodes and the localization error obtained from various localization services.

Impact of Synchronization Accuracy

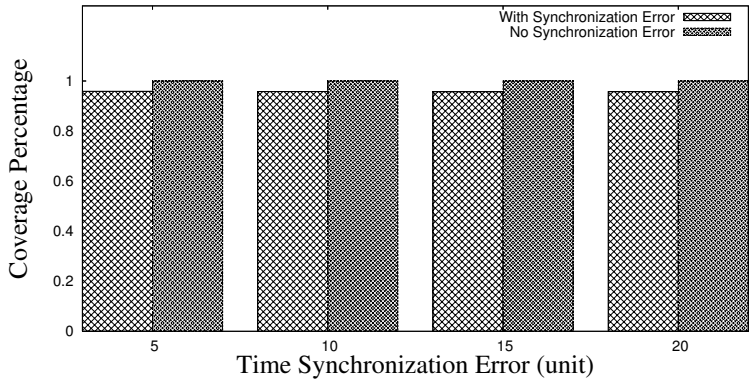


Figure 2.40: Sensing Coverage vs. Synchronization Error (Density = 2)

In this experiment, we study the impact of synchronization accuracy on the uScan protocol. Figure 2.40 shows under the full coverage mode, the coverage percentage of active sensor nodes with the synchronization error varies from 5 to 20 units of time (one duration of an active bit in the schedule bits S). From this Figure, we can see under all levels of synchronization errors, the quality of sensing coverage is rarely degraded. For example, with the largest synchronization error of 20 units of time measured in the experiment, the uScan still can provide an average sensing coverage of 95.74 percentage during the network operation. In many sensor network applications, such a degree of degradation is tolerable. In addition, synchronization schemes, such as FTSP [29], are already capable of providing synchronization accuracy at tens of microseconds. Both factors indicate that the uScan protocol can be applied in the realistic deployment.

Impact of Localization Errors

Figure 2.41 shows the average single node lifetime under different network node densities and node localization errors in the full coverage mode. In this experiment, we compare the single node lifetime in the network under scenarios including no localization errors, the localization error is within five meters and the localization error is within ten meters. From the Figure, we can see the average single node lifetime has little variations in all three scenarios. The largest single node lifetime reduction we observe is from 5.938 to 5.918, which is about 0.34% to the original single node lifetime with no localization

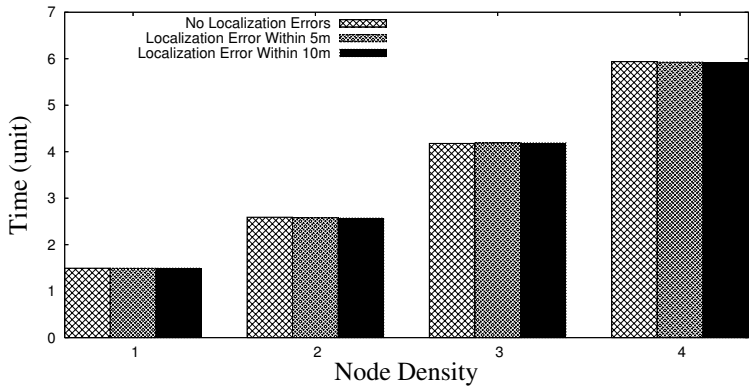


Figure 2.41: Avg. Single Node Life vs. Node Densities

error, when the node density is 4 and the localization error is within ten meters. Since many localization schemes (e.g., SpotLight [31], StarDust [32] and methods proposed in [33] by Patwari et al.) can provide meter-level localization accuracy, we believe uScan is compatible with these existing localization services.

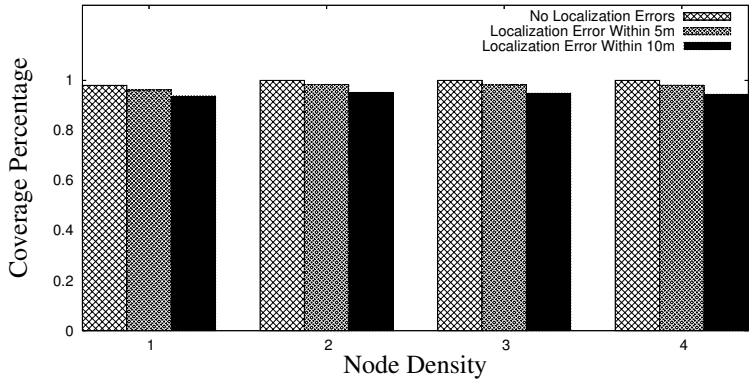


Figure 2.42: Sensing Coverage vs. Node Densities

In addition, in Figure 2.42, we study the sensing coverage under different node densities and localization errors in the full coverage mode. From the Figure, we can see for both localization errors within five meters and ten meters, the quality of sensing coverage remains at a rather high level. For example, the worst degree of coverage in the experiment occurs when the node density is four and the localization error is within ten meters. However, even under such conditions, uScan can still maintain a 94.4% of

coverage. According to above results, we conclude that our proposed uScan is robust under moderate localization errors.

2.8 Related work

Physical sensing coverage is the research focus in the beginning. In [7], authors support full surveillance coverage based on an off-duty eligibility rule. DiffSurv [8] provides differentiated surveillance to an area with a certain degree of coverage. In [28], surveillance coverage is achieved through probing. Several other works focus more on the theoretical results of sensing coverage. Kumar et al. [5] identify a critical bound for k-coverage in a network, assuming a node is randomly turned on with a certain probability. In [34], Kumar et al. investigate the k-barrier coverage problem, identifying the critical condition for weak k-barrier coverage. Several algorithms are designed based on the concept of set cover. In [4], Cardei et al. propose two heuristic algorithms to identify a maximum number of set covers to monitor a set of static targets at known locations. In [3], Abrams et al. propose three approximation algorithms for a relaxed version of the previously defined SET K-COVER problem [6].

The integration between coverage and communication becomes an important topic recently. The coordinated sleep algorithm proposed in RACP [12] provides continuous coverage and compares the approach with a random sleep algorithm, revealing the tradeoffs between energy savings and required redundancy to achieve desired delays. The Coverage Configuration Protocol (CCP) [14] provides an energy-efficient sensing coverage, which is integrated with SPAN for connectivity. In [13], Shakkottai et al. validate that even with high node unreliability and low transmission power, the connectivity with coverage can be maintained in a dense grid network. In [10], using proposed Markov model of a network, Chiasserini trades off between energy saving and network performance, when node switches between on/off state. In [11], according to the desired sensing coverage, Choi and Das select certain nodes in each sensing round to provide the coverage and necessary extra nodes to reduce the transmission delay. Zhang and Hou in [15] designed a decentralized density control algorithm, called Optimal Geographical Density Control (OGDC), which can maintain coverage as well as connectivity, regardless of the relationship between the radio range and the sensing range. In [16], Liu et al.

propose a joint scheduling for coverage and connectivity, with randomized scheduling to provide statistical sensing coverage and then switch on extra sensors, if necessary, for network connectivity.

To achieve a higher energy efficiency, several recent works focus on the partial coverage within a fixed time delay. In [9], nodes coordinate to guarantee the worst-case detection delay and to minimize the average detection delay. Another type of temporal guarantee is to analyze the detection delay (stealth distance) in the context of tracking. In [17, 18, 30], they assume the network is partially covered and provide a theoretical analysis and simulation on the delay (or stealth distance) before a target is detected.

Our work is unique in the following aspects: (i) uSense is a unified architecture instead of an individual solution. We are the first to propose the concept of generic switching. (ii) uScan demonstrates a novel two-level global scheduling method that can significantly reduce energy consumption. (iii) Our set-cover is uniquely implemented at the second level, allowing the first-level optimization. For example, we demonstrate optimal cover sets can be obtained in linear time when the coverage area is a continuous curve. In contrast, single-level set-cover approaches [4, 3] could be inefficient under non-uniform node distribution.

2.9 Conclusion

In this work, we propose a unified sensing architecture called uSense. It features an asymmetric design, which supports various kinds of coverage algorithms with a simple generic switching algorithm in sensor nodes. It allows us to flexibly change coverage algorithms with only two parameters. Another major contribution of this work is a two-level global scheduling algorithm called uScan, which is seamlessly supported by the uSense architecture. In the first level, we propose an optimal scheduling algorithm in terms of minimizing area breach. In the second level, we propose a linear algorithm to address the set-cover problem when the layout of tiles satisfies certain conditions. We have invested significant effort to evaluate our design, which includes a network of 30 MicaZ motes, an extensive simulation with 10,000 nodes, as well as theoretical analysis. We also study the practical issues such as the impact of the time synchronization and

localization errors to the performance of uScan. With three novel ideas: *Asymmetric Architecture*, *Generic Switching* and *Global Scheduling*, our work has successfully achieved flexibility and efficiency for the sensor network coverage problem.

Chapter 3

Data Forwarding in Low-Duty-Cycle Sensor Networks

In extremely low duty-cycle sensor networks, end-to-end communications cannot afford to maintain an always-awake communication backbone. Low duty-cycle, accompanied by the unreliable nature of wireless communication, makes it essential to design a new data forwarding scheme for such networks, so as to achieve network energy efficiency, reliability, and timeliness in an integrated fashion.

In this chapter, we introduce the concept of Dynamic Switch-based Forwarding (DSF) that optimizes the (i) expected data delivery ratio, (ii) expected communication delay, or (iii) expected energy consumption. DSF is designed for networks with possibly unreliable communication links and predetermined node communication schedules. Interestingly, we reveal that allowing *opportunistic looping* can actually reduce the end-to-end delay. To our knowledge, these are the most encouraging results to date in this new research direction. In this chapter, DSF is evaluated with a theoretical analysis, extensive simulation, and physical testbed consisting of 20 MicaZ motes. Results reveal the remarkable advantage of DSF in extremely low duty-cycle sensor networks in comparison to three well-known solutions (ETX [35], PRR×D [36] and DESS [37]). We also demonstrate our solution defaults into ETX in always-awake networks and DESS in perfect-link networks.

3.1 Introduction

Many sensor network applications need to last a long time with a limited energy supply. To resolve the conflict between limited energy and application lifetime requirements, it is necessary to reduce node communication and sensing duty cycles. With the growing gap between application requirements and the slow progress in battery capacity [38], there are an increasing number of extremely low duty-cycle sensor networks designed and deployed. Together with lossy radio links, these new networks impose new challenges for data forwarding protocols.

In this work, we focus on *extremely low duty-cycle sensor networks with unreliable communication links*, in which energy management protocols [39, 40, 14, 8] schedule sensing and communication at each individual sensor device to enable a duty cycle of 1% or less. Essentially, during the operation of sensor applications, sensor nodes activate very briefly and stay in a dormant state for a very long period of time. Due to the devices' extremely limited energy budget, maintaining an always-awake communication backbone becomes infeasible. Consequently to forward a packet, a sender may experience *sleep latency* – the time spent waiting for the receiver to wake up.

In this chapter we attempt to design a new data delivery method to optimize source-to-sink data delivery ratio, end-to-end (E2E) delay, or energy consumption under *unreliable* and *intermittent* connectivity within scheduled networks. The major intellectual contributions of this work are as follows:

- We propose a simple yet effective representation of sensor working schedules, which for the first time enables us to reveal the time-dependent data forwarding behavior within extremely low duty-cycle sensor networks.
- To the best of our knowledge, this is the first in-depth work to investigate the combined effect of *sleep latency* and *unreliable communication links*, which dramatically reduces the effectiveness of the existing solutions. A novel dynamic switch-based forwarding technique over time-dependent networks is proposed to achieve optimal expected delivery ratio (EDR), expected E2E delay (EED), or expected energy consumption (EEC), respectively.

- Interestingly, we demonstrate that the allowance of certain data forwarding loops can actually reduce end-to-end data delivery delay. In addition, we address practical issues such as time synchronization and changes in link quality over time.

The rest of the chapter is organized as follows: Section 3.2 presents the related work. Section 3.3 describes the need for a new data forwarding technique in extremely low duty-cycle sensor networks. Section 3.4 articulates the network model and related assumptions. Section 3.5 introduces the detailed design of DSF and discusses related issues. Section 3.6 describes our system implementation and provides an evaluation on the TinyOS/Mote platform. Simulation results are presented in Section 3.7. Section 3.8 concludes the chapter.

3.2 Related Work

The contribution of our work lies in the intersection of two important cutting-edge research topics. We demonstrate that the intriguing interaction between unreliable links and low-power duty-cycling necessitates a fundamentally new approach.

Link-Quality-Based Forwarding: Many recent works [41, 42, 43] reveal that wireless communication links, especially for the low-power sensor devices, are extremely unreliable and have a significant impact on data delivery.

De Couto et al. introduce the expected transmission count metric (ETX) to find high-throughput paths on multi-hop wireless networks [35]. Woo et al. show that cost-based routing using a minimum expected transmission metric obtains good performance in wireless sensor networks [44]. Seada et al. study the distance-hop trade-off for geographic routing in wireless sensor networks and show that the product of the packet reception rate (PRR) and the distance traversed toward the destination (D) is an optimal metric (PRR$\times$D) for selecting a next-hop forwarder [36]. Lee et al. present SOFA, an on-demand solicitation-based forwarding protocol and show that SOFA outperforms the commonly used link estimation-based routing schemes implemented in TinyOS [45].

In these works, the authors assume the constant availability of connectivity with no sleep latency, which may not be true in extremely low duty-cycle sensor networks.

Sleep-Latency-Based Forwarding: In the research direction of low duty-cycle networks, Dousse et al. provide a solid analysis of bounds of the delay for sending data from a node to a sink in the networks with completely uncoordinated node working schedules [46]. Cao et al. present a Streamlined Forwarding (SF) strategy to reduce delay in end-to-end communication [9]. This strategy works well under a constrained communication pattern where only one sink is allowed. Yu et al. present algorithms to minimize the overall energy dissipation of the sensor nodes in the network subject to the latency constraint [47]. Lu et al. introduce various techniques for minimizing communication latency while providing energy-efficient periodic sleep cycles for nodes in wireless sensor networks [37]. More recently, Keshavarzian et al. introduce a multi-parent forwarding technique and propose a heuristic algorithm for assigning parents to the nodes in the network [48]. We note, however, that all these approaches in low duty-cycle networking assume perfect communication links.

We note that many MAC protocols, such as B-MAC [49], S-MAC [50], FPA [51] and SCP-MAC [52], effectively deal with the issues of lossy radio links through FEC/ARQ and reduce duty-cycle through the Low-Power-Listening (LPL) [49]. These intelligent layer 2 protocols use implicit network information, such as packet transmissions, in order to optimize their underlying schedules or energy use. In this chapter, we consider the dual of this problem by using information from layer 2 at the network layer to make better link selections.

To the best of our knowledge, no prior work has thoroughly studied the impact of both *lossy radio links* and *sleep latency* at the network layer. In this work, we reveal that these two issues are intrinsically correlated and that a new forwarding protocol can benefit from considering both.

3.3 Motivation

Our work is motivated by the interesting intersection between sleep latency and unreliable communication links in wireless sensor networks.

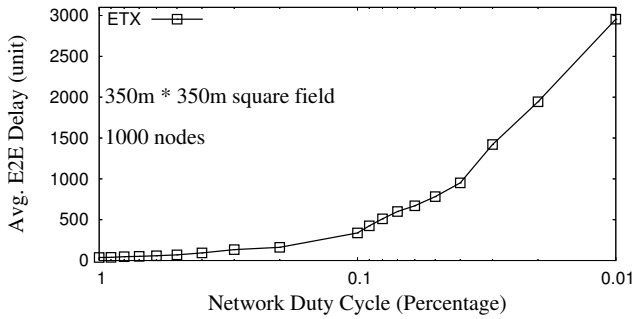


Figure 3.1: E2E Delay vs. Network Duty Cycle

First, the state-of-the-art link-quality-based forwarding strategies such as ETX [35] and PRR×D [36] have demonstrated their superiority at improving network throughput and communication delay in traditional ad hoc and sensor networks. For both ETX and PRR×D, during a certain period of time each node usually has one fixed forwarding node for a destination. However, in extremely low duty-cycle scheduled sensor networks, metrics such as the expected transmission count (ETX) would suffer excessive delivery delays when waiting for the fixed receiver to wake up again if the ongoing packet transmission fails. Figure 3.1 shows the E2E delays from a randomly chosen source node to the sink node using ETX forwarding metrics under different network duty cycles in a randomly-generated network topology. The simulation was repeated 1000 times and the average value is reported in Figure 3.1 (in log-scale), which shows that as network duty cycle decreases, the E2E delay grows significantly. For example, at the duty cycle of 100%, the E2E delay of ETX is only 37.6 units of time. In contrast, when the duty cycle drops to 1%, the E2E delay increases to 2955.5 units of time, which is approximately an 80-fold performance degradation in end-to-end delay!

Second, sleep-latency-based forwarding [9, 37] ignores the reality that wireless radio quality is highly unreliable and that thus the optimality of their approaches holds only when the link quality in the network is perfect. Figure 3.2 shows the E2E delay from a randomly chosen source node to the sink node using delivery methods proposed in [37]

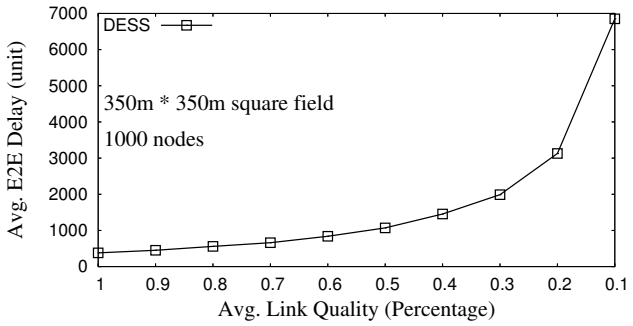


Figure 3.2: E2E Delay vs. Average Link Quality

under different average link quality in a random generated network topology. As shown in the figure, the E2E delay increases from 380.0 to 6851.4 units of time while the average link quality decreases from 100% to 10%, which is approximate a 20-fold performance degradation, even though global scheduling information is available.

The main observation from our initial studies is that both the link quality and the duty cycle of sensor nodes can significantly impact end-to-end communication. Although link-quality-based forwarding [35, 36] and sleep-latency-based forwarding [9, 37] have demonstrated their effectiveness in their own contexts, they fail to deal with the combined effect exhibited in many real-world sensor network applications. This limitation motivates us to design a new data forwarding technique, which we discuss in the rest of the chapter.

3.4 Models and Assumptions

Before presenting DSF in detail, we present the network model and assumptions used in this work. To simplify our description, we introduce DSF's design in a synchronized mode with discrete time. Later on, we explain why DSF works without time slots and only requires local synchronization. In other words, DSF works in CSMA networks where nodes are duty-cycled by upper-layer protocols such as sensing coverage [39] and power management [48].

3.4.1 Network Model

We assume a network with N sensor nodes. At a given point of time t a sensor node is in either an active or a dormant state. When a node is in the active state, it can sense and receive packets transmitted from neighboring nodes. When a node is in the dormant state, it turns off all function modules except a timer (for the purpose of waking itself up). In other words, a node can wake up to transmit a packet at any time, but can receive packets only when it is in its active state. Formally, we denote the network status at time t as $G(t) = (V, E(t))$, where V is a complete set of N nodes within the network, and $E(t)$ is a set of directed edges at time t. An edge $e(i, j)$ belongs to $E(t)$ if and only if (1) node n_i is a neighboring node of n_j, and (2) n_j is active and hence able to receive data at time t. Essentially, $G(t)$ represents the potential traffic flow within the network at time t. Obviously the connectivity of $G(t)$ varies with time. In other words, $G(t)$ is a time-dependent network.

We represent the states of each node n_i with a working schedule $\Gamma_i = (\omega_i, \tau)$.

- ω_i is an infinite binary string, in which 1 denotes the active state and 0 denotes the dormant state. Clearly, the duty cycle of a node is the percentage of 1's in the binary string. Since the working schedules of the sensor nodes are normally periodic (for sensing purposes), the infinite binary string ω_i can be described using a regular expression.

- The state transitions between active and inactive states are time-driven. We use τ to denote the time span a bit in the binary string ω_i. For example, the total time-span of the binary string (1001) with τ of 2 seconds is 8 seconds.

We note that the simple 2-tuple (ω_i, τ) is generic enough to represent arbitrary sensor nodes working schedules. Theoretically, when $\tau \rightarrow 0$, ω_i can precisely characterize any on/off behavior of node n_i. For clarity of presentation, we begin our design with a simplified assumption that it takes time τ to transmit one packet and receive acknowledgment from a receiver. The assumption on the round-trip transmission time bound τ holds well when traffic/congestion is low, which is the case in extremely low duty-cycle sensor networks. In addition, B-MAC [49] has already used link-level implicit acknowledgement to support fixed round-trip transmission time. We address the case that time τ is long enough to transmit multiple packets in Section 3.5.5.

3.4.2 Time-Expanded Network

To visualize the data delivery process in a time-dependent network $G(t) = (V, E(t))$, we replicate $G(t)$ with regular graphs $G = (V, E)$ along with the time dimension. We call this is a *time-expanded network*. In this section, for a given sensor network topology and node working schedules, we describe how we can build a corresponding time-expanded network. The resulting time-expanded network can help us better understand the data delivery method introduced in the rest of the chapter.

Given a network $G(t) = (V, E(t))$ with n nodes and node working schedules $\Gamma_i = (\omega_i, \tau)$, where $i \in V$, we use the following rules to construct its corresponding time-expanded network.

- For any node $i \in V$ at time t, we build a distinct node N_{it}.
- For each newly built node N_{it}, if node j is a neighboring node of the node i and p is the position of first active bit in ω_j after time t, we build a directed edge from N_{it} to N_{jp} with a length of $(p - t)\tau$.
- At the destination node d, we connect all its time-expanded nodes to a null node with edge lengths of zero.

To illustrate the above network mapping rules, we provide a walk-through of time-expanded network construction from a time dependent graph. Figure 3.4 shows how to construct a time-expanded network from the linear time-dependent network shown in

Figure 3.3. As shown in Figure 3.3, for node 1 at time 1, the only node that is within its communication range is node 2, and its first active state after time 1 appears at time 3, so in Figure 3.4, we build a directed edge from node 1 at time 1 to node 2 at time 3 with an edge length of 2τ. Similarly, we construct other edges in Figure 3.4. Finally, for the destination node 4, we connect all its time-expanded nodes from time 1 to time 6 to a null node with edge lengths of zero.

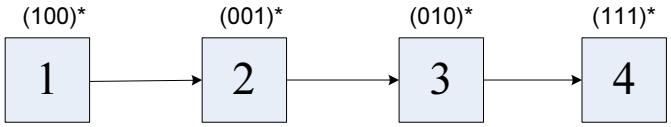


Figure 3.3: A Linear Network

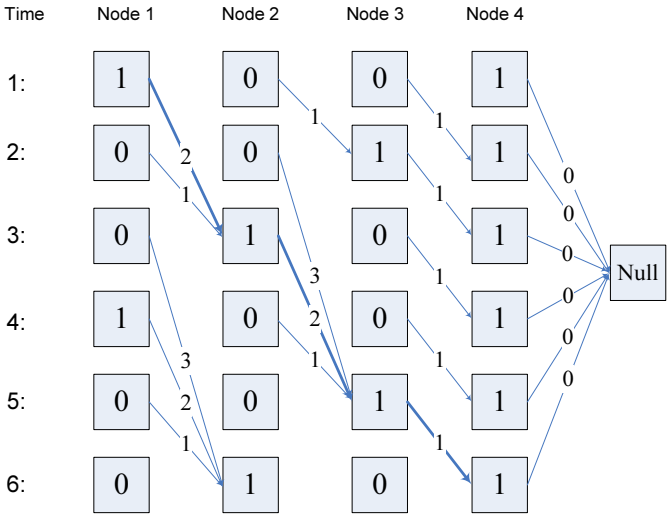


Figure 3.4: Time-Expanded Network

3.4.3 E2E Delay in Time-Expanded Network

Obviously, if all the nodes are in active states, end-to-end (E2E) delay in the above network model equals $H\tau$, where H is the minimum number of hops between a source and a destination. However, if nodes in a network have certain working schedules $\Gamma = \{\Gamma_1, \Gamma_2, ..., \Gamma_N\}$, transmission at each hop could be delayed by waiting for the intermediate receivers to wake up.

To further illustrate the data delivery process in the extremely low duty-cycle network, Figure 3.4 demonstrates the process of delivering a packet from node 1 to node 4 when the packet is ready to be sent at time 1. For the sake of simplicity, in this example we assume all links are perfect with no packet loss and will discuss cases when the link quality is not perfect in detail in the following sections. At the first two time intervals, node 2 (the only neighbor of node 1) is in the dormant state, and thus no packets could be transmitted. At the third time interval, node 2 becomes active, which allows node 1 to transmit a packet to it, so the packet is delivered from node 1 to node 2. At the second hop, node 2 waits one time interval for node 3 to wake up, and the packet arrives at node 3 at time 5. Finally, since node 4 is active all the time, without any additional waiting, node 3 delivers the packet to node 4 at time 6. The total end-to-end delay therefore is 5 units of time (i.e., arrives at the destination at time 6 and is ready to be sent at time 1 at the source).

3.5 Main Design

As shown in Section 3.4.3, when the link quality is perfect, the end-to-end delay is the sum of two types of delays: (1) the total transmission delay, which is the product of number of hops and τ, and (2) The sleep latency, which is the time spent on waiting for the receivers to wake up at each hop. However, the unreliable radio links between low-power sensor devices suggests that the packet transmission between a sender and a receiver would not always be 100% successful. As a result, the waiting time at each hop is highly impacted not only by the node working schedule but also by the link quality, which inspires us to design a dynamic switch-based data forwarding protocol.

Since every operation within an extremely low duty-cycle sensor network is time-dependent, for the sake of clarity we use the terms *node* and *time-expand node* interchangeably in the rest of the chapter. We have organized this design section into five components. Section 3.5.1 describes the basic design of Dynamic Switch-based Forwarding (DSF). Section 3.5.2 analyzes the expected delivery ratio, E2E delay, and energy consumption, assuming the forwarding action is known *a priori*. Section 3.5.3 optimizes the forwarding action to achieve maximum delivery ratio, minimal delay, and energy efficiency, respectively. Section 3.5.4 describes the distributed implementation of DSF. Section 3.5.5 presents our observations and extensions of DSF.

3.5.1 The Basic Design of DSF

Differently than traditional data forwarding techniques such as ETX and PRR×D, we allow *multiple potential forwarding nodes* at each hop. For a given sink, each node maintains a sequence of forwarding nodes sorted in the order of the wake-up time associated with them. To start sending a packet, a node looks up the time associated with the first node in the sequence, wakes up at that time interval, and tries to send the packet. If the transmission is successful, forwarding is done. Otherwise, the node fetches the next wake-up time from the sequence and tries to send the packet again. This retransmission process over a single hop continues until the sending node confirms that the packet has been successfully received by one of forwarding nodes or the sending node reaches the end of the sequence and drops the packet.

Formally, we define the sequence of forwarding nodes at a node e as: S_n^e

Definition 2 (Forwarding Sequence S_n^e) *S_n^e is a sequence of n nodes that can forward packets from node e to the sink. This sequence is sorted based on the wake-up time of the nodes. Formally, $S_n^e = (s_1^e, s_2^e, \cdots, s_n^e)$. Let $t(s_i^e)$ be the wake-up time of node s_i^e, Forwarding sequence S^e satisfies $t(e) < t(s_1^e) < t(s_2^e) < \cdots < t(s_n^e)$.*

Figure 3.5 demonstrates the packet transmission process between one sender and n nodes in its forwarding sequence. In Figure 3.5, node A has a packet to be sent and its forwarding sequence is $S_n^A = (B_1, B_2, \cdots, B_n)$. First, node A wakes up at time t_1 and tries to transmit the packet to the node B_1. If the data delivery is successful, node A ends the current packet forwarding session. However, if the transmission fails, the node A wakes up again at time t_2 and tries to send the packet to the node B_2. This retransmission process continues with node A repeatedly trying to send the packet to the node in the sequence S_n^A. If the transmission fails at the last node B_n, node A drops the packet.

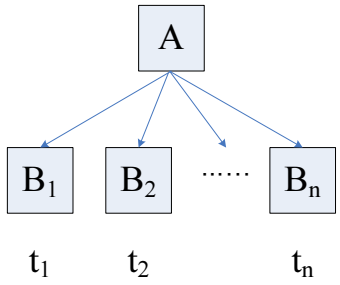


Figure 3.5: Example of Dynamic Switching

From the above example, we can see that the major advantage of dynamic switching is the use of a forwarding sequence to reduce the time spent on transmitting a packet successfully at each hop rather than waiting for a particular forwarding node to wake up again after failure, as in such solutions as ETX, PRR×D and DESS.

3.5.2 The Modeling of EDR, EED, and EEC

Given a known forwarding sequence S_n^e at a node e , we can model the expected delivery ratio, the expected E2E delay and the expected energy consumption for the node. Here,

for the sake of clarity, we describe a scenario with a single sink node, that can be extended easily for scenarios with multiple sink nodes.

Formally, these three metrics are defined as:

Definition 3 (Expected Delivery Ratio $EDR_e(S_n^e)$) *The expected delivery ratio at node e for a given forwarding sequence S_n^e, denoted by $EDR_e(S_n^e)$, is the expected packet delivery ratio from node e to the sink node (over multi-hop path).*

Definition 4 (Expected E2E Delay $EED_e(S_n^e)$) *The expected E2E Delay at node e for a given forwarding sequence S_n^e, denoted by $EED_e(S_n^e)$, is the expected data delivery delay for the packets sent by node e and received by the sink node (over multi-hop path).*

Definition 5 (Expected Energy Consumption $EEC_e(S_n^e)$) *The expected energy consumption at node e for a given forwarding sequence S_n^e, denoted by $EEC_e(S_n^e)$, is the expected energy consumption to deliver a packet from node e to the sink node (over multi-hop path). We note that since receiving (idle) energy is fixed for a given working schedule, we include only senders' transmission energy in EEC.*

Our model for computing EDR, EED, and EEC values is distributed and can be executed at individual sensor nodes independently. At the sink node (b), obviously, its forwarding sequence is empty, the $EDR_b(\emptyset)$ value is 100% (i.e., no packet loss), while $EED_b(\emptyset)$ and $EEC_b(\emptyset)$ values are both zeros (i.e., no delay and no energy consumption). Consequently, we can obtain following initial equations:

$$EDR_b(\emptyset) = 1 \quad , \quad EED_b(\emptyset) = 0 \quad , \quad EEC_b(\emptyset) = 0 \tag{3.1}$$

Let the bi-directional link quality p_{ei} denotes the success ratio of a round-trip transmission (DATA and ACK) between node e and the i^{th} forwarder in S_n^e. The link quality p_{ei} can be influenced by multiple factors such as transmission power and the distance between a sender and a receiver. We note that in extremely low duty-cycle sensor networks, traffic congestion is rare and hence has little effect on link quality.

The overall probability $P^e(i)$ that a packet transmission by node e is successful at the i^{th} forwarder (after $i-1$ failures) can be represented as:

$$P^e(i) = [\prod_{j=1}^{i-1}(1-p_{ej})]p_{ei} \tag{3.2}$$

Expected Delivery Ratio (EDR): Obviously, EDR value for node e is the sum of the product of the probability that the transmission is successful at a particular forwarder and its corresponding EDR value for all nodes in S_n^e. Assuming node e has n nodes in its forwarding sequence and letting EDR_i be the EDR value for the i^{th} forwarder (s_i^e) in S_n^e, we have the following recursive equation for $EDR_e(S_n^e)$.

$$EDR_e(S_n^e) = \sum_{i=1}^{n} P^e(i)EDR_i \tag{3.3}$$

Expected E2E Delay (EED): The EED value of node e represents the expected delay for the packets sent by node e that reach the sink node b. Consequently, the probability that the packet transmission is successful at a certain forwarder is under the condition that the packet is delivered by one of the forwarders in S_n^e. Therefore, the conditional probability is $P^e(i)' = \frac{P^e(i)EDR_i}{EDR_e(S_n^e)}$. Letting EED_i be the EED value for the i^{th} forwarder in node e's forwarding sequence and d_i be the delay for node e to wait node s_i^e in S_n^e to wake up, then $EED_b(S_n^e)$ can be represented as:

$$EED_e(S_n^e) = \textstyle\sum_{i=1}^{n} P^e(i)'(d_i + EED_i) \tag{3.4}$$

Expected Energy Consumption(EEC): Similarly, let EEC_i be the EEC value for the i^{th} forwarder in S_n^e and that the expected energy consumption for successful packet transmission at node s_i^e is the sum of EEC_i and i units of energy consumption (note that energy wasted in $i-1$ failed transmission should be included as well). The probability that the retransmission of a packet reaches the i^{th} forwarder $P^e(i)'$ at node e is conditional on the data delivery ratio $EDR_e(S_n^e)$. Therefore, $P^e(i)' = \frac{P^e(i)EDR_i}{EDR_e(S_n^e)}$, and we can formulate the $EEC_e(S_n^e)$ as:

$$EEC_e(S_n^e) = \textstyle\sum_{i=1}^{n} P^e(i)'(i + EEC_i) \tag{3.5}$$

The recursive calculation of EDR, EED and EEC can be implemented at individual nodes distributively. The main idea is to radiate known initial conditions ($EDR_b(\emptyset) = 1$,

$EED_b(\emptyset) = 0$,$EEC_b(\emptyset) = 0$) from the sink node, so that the process of calculating EDR, EED and EEC values propagates outward from the sink nodes to the rest of the network.

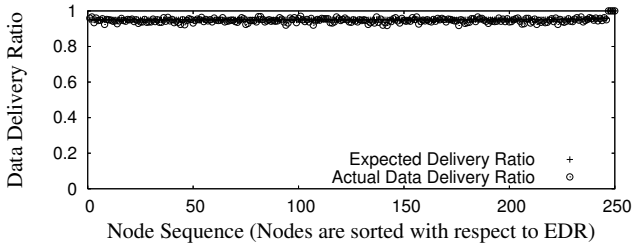


Figure 3.6: Expected Delivery Ratio vs. Actual Delivery Ratio

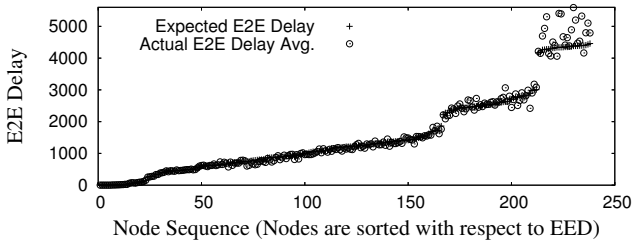


Figure 3.7: Expected E2E Delay vs. Actual E2E Delay

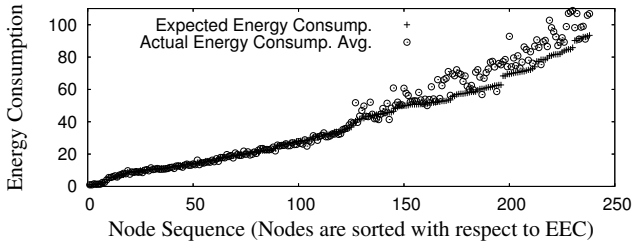


Figure 3.8: Expected Energy Consumption vs. Actual Energy Consumption

Model Validation: Figures 3.6, 3.7, and 3.8 show a pair-wise comparison of the calculated expectation values and the simulated E2E delay and energy consumption according to the radio model in [53], which considers the oscillation of radio links, for a randomly generated network graph in which each node has an average of six neighbors. In all three figures, the nodes are sorted with respect to their corresponding expectation values and we simulate the packet delivery process to the sink node 1,000 times and report the average value. For the data delivery ratio, the average difference between expectation and simulation data is 0.96%, with a standard deviation of 0.74%.

For the E2E delay, we observe an average 5.5% difference between expectation and simulation results, with a standard deviation of 4.8%. For the transmission energy consumption, the difference is an average of 6.1%, with a standard deviation of 4.7%. We also studied the deviation between expectation values and simulation results for many other randomly generated network graphs and obtained similar results. Figures 3.6, 3.7, and 3.8 confirm that our model for computing expected delivery ratio, E2E delay, and energy consumption accurately captures the behavior of the packet delivery process in DSF.

3.5.3 Optimizing the Forwarding Sequence

In the previous section, we described the model for calculating EDR, EED, and EEC for a *given* forwarding sequence. In this section, we will discuss how we can obtain a forwarding sequence that is *optimal* in terms of the maximum expected data delivery ratio, minimum expected E2E delay, or minimum expected transmission energy consumption at individual sensor nodes, respectively.

In practical network settings, especially in low duty-cycle sensor networks, a sender should not endlessly retransmit a packet because it would consume significant energy at the sending nodes. Therefore, we set the maximum time bound for a sender to retransmit a particular packet as T. Consequently, at node e, with known neighboring nodes and their corresponding working schedule Γ, we can have a full sequence of potential forwarding nodes that wake up before T.

Formally, let s_i^e be a next-hop node with wake-up time $t(s_i^e)$. Node e's full sequence S_m^e under the bound T is:

$$S_m^e = (s_1^e, s_2^e, \cdots, s_m^e) \;\; where \;\; s \in S_m^e \iff t(e) \leq t(s) \leq t(e) + T.$$

While noting that it is possible that multiple next-hop nodes may wake up at the same time, for now we first describe a simplified scenario where no next-hop nodes wake up at the same time interval. The revised solution for accommodating multiple wake-up nodes is presented in Section 3.5.5.

Optimizing Expected Delivery Ratio (EDR)

Because the length of the potential forwarding sequence of a node is finitely subject to the maximum retransmission time interval T, under the reality of unreliable link quality among pairs of wireless sensor devices, packets sent by a source node may not all arrive the destination sink node. Therefore, when reliable transmission has the highest priority for a sensor network application, the optimization of the expected data delivery ratio (EDR) is critical.

Intuitively, in order to maximize the expected data delivery ratio at node e, we should try to send packets as long as one of the next-hop nodes is awake. The reasoning behind this is plausible, as since we want to maximize the expected data delivery ratio, we should take every opportunity to move the packet out of the sender. However, this intuition does not lead us to an optimal expected data delivery ratio, and Figure 3.9 presents a counterexample. In Figure 3.9, suppose the full forwarding sequence of the node S is $S_2^S = (A, B)$. If we choose both node A and B to form S's forwarding sequence S_2^S, according to the Equation 3.3, $EDR_e(S_2^S) = 10\%$. In contrast, if we choose only node B to be included in S_1^S, the corresponding $EDR_e(S_1^S) = 100\%$. Therefore, in order to optimize the expected data delivery ratio at a node e, we shall select a subsequence from the full sequence S_m^e. By definition, a subsequence can be obtained by removing some of the elements from the original sequence without disturbing the relative positions of the remaining elements. As an example, (B, E, D, G) is a subsequence of (A, B, E, C, F, D, H, G).

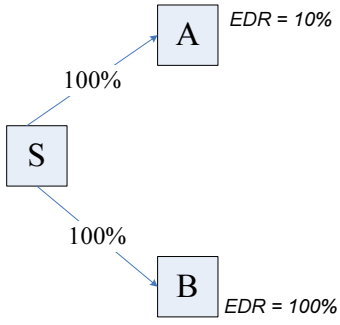


Figure 3.9: Example for Selecting a Subset of Nodes in Potential Forwarding Sequence

To select an optimal subsequence S_{opt}^e from the full sequence S_m^e, we adopt a dynamic programming approach. Clearly, the last node s_m^e in S_m^e must be included in S_{opt}^e, since

s_m^e provides the last chance for node e to retransmit before the packet is dropped. Starting from this optimal substructure, we can attempt to include nodes (one by one) from S_m^e ***backwardly*** into S_{opt}^e. If the inclusion of a node from S_m^e into S_{opt}^e increases $EDR_e(S_{opt}^e)$, we then add this node into S_{opt}^e permanently. Otherwise, we discard the node and try to add the next node. The above forwarding sequence selection process continues until we reach the node s_1^e in the full sequence S_m^e. The optimality of this dynamic programming algorithm is based on the fact that the optimal $EDR_e(S_{opt}^e)$ can be constructed efficiently from its optimal substructures. *The decisions made to include or exclude a* ***later*** *node in the forwarding sequence does not affect the optimality of decisions made to include or exclude* ***earlier*** *nodes and vice versa.* For each backward augmentation of the forwarding sequence, we guarantee the maximum data delivery ratio of the sequence between the newly augmented node and the last node. This forwarding sequence, then, serves as an optimal substructure for augmenting additional forwarders until the process reaches the first node in the sequence.

Let $S_{opt}^e(k)$ denote the optimal forwarding subsequence in terms of maximizing EDR metric from the sequence $S_k^e = (s_{m-k+1}^e, s_{m-k+2}^e, \ldots, s_m^e)$. Obviously, $S_{opt}^e(m)$ is the optimal subsequence we want to obtain.

We have the following initial optimal substructure:

$$\begin{aligned} S_{opt}^e(0) &= () \\ S_{opt}^e(1) &= (s_m^e) \end{aligned} \tag{3.6}$$

Building upon the previous optimal substructure, when we attempt to include the next node s_j^e in S_m^e into $S_{opt}^e(k-1)$ backwardly, there are two possible outcomes:

- According to the model of $EDR_e(S^e)$, if the appending of node s_j^e to $S_{opt}^e(k-1)$ increases the expected delivery ratio, we insert node s_j^e in front of the existing sequence $S_{opt}^e(k-1)$ to obtain $S_{opt}^e(k)$.
- If the inclusion of node s_j^e into sequence $S_{opt}^e(k-1)$ does not increase the data delivery ratio, the optimal forwarding sequence remains unchanged.

Formally, let $s_j^e \oplus S$ denote inserting node s_j^e to the front of the sequence S, the corresponding recursive equation for $S_{opt}^e(k)$ can be represented as:

$$S_{opt}^e(k) = \begin{cases} S_{opt}^e(k-1) & EDR_e(S_{opt}^e(k-1)) > EDR_e(s_j^e \oplus S_{opt}^e(k-1)) \\ s_j^e \oplus S_{opt}^e(k-1) & Otherwise \end{cases} \tag{3.7}$$

The complexity for this dynamic programming algorithm is $\mathcal{O}(mT)$, where m is the density of next-hop nodes and T is the maximum per-hop delay allowed.

Optimizing Expected E2E Delay (EED)

In many sensor network applications, such as military surveillance [54], target tracking [55] and infrastructure monitoring [56], the delay for the source-to-sink communication is critical to the performance of the system.

We note that if there is no bound on the expected delivery ratio (EDR) for the forwarding sequence, the optimal forwarding sequence in terms of minimizing delay can be trivially achieved by including only a single node j which has the minimum $(d_j + EED_j)$ value among all nodes in S_m^e (Equation 3.4). However, with such a quick-and-dirty solution, especially when the link quality between node e and node j is low, node e may suffer from an extremely low packet delivery ratio to the sink node and consequently may cause the whole network to be unavailable. Therefore, it is important to minimize the EED metric for the node e under the constraint that the EDR metric of the forwarding sequence is greater than a certain bound R. The bound R must be less or equal to the optimal EDR value that could be achieved at the node e.

Similarly to maximizing EDR, we also adopt a dynamic programming approach to select a subset of nodes in S_m^e *backwardly* to optimize EED. But in contrast, the last node in S_m^e is no longer guaranteed to be the optimal initial optimal substructure, since the inclusion of the node may increase the expected E2E delay. Instead, to optimize EED, we need to try every node in the full sequence S_m^e as the last node in the optimal subsequence. For example, if we suppose $S_m^e = (B, E, D, G)$, we need to obtain optimal subsequences from (B, E, D, G), (B, E, D), (B, E), and (B) with G, D, E, B chosen, respectively.

Suppose node s_{last}^e is selected as the last node and $S_{opt}^e(last, k)$ represents the optimal forwarding subsequence in terms of EED chosen from the sequence $S_k^e(last) = (s_{last-k+1}^e, s_{last-k+2}^e, \ldots, s_{last}^e)$, where $k \leq last$ and $last \in \{1, 2, \ldots, m\}$.

For each last node, we have the following initial optimal substructure for $S_{opt}^e(last, k)$ in terms of minimal EED:

$$\begin{aligned} S_{opt}^e(last, 0) &= () \\ S_{opt}^e(last, 1) &= (s_{last}^e) \end{aligned} \tag{3.8}$$

Similar to the recursive equations for maximizing EDR, for each node s_j^e in $S_k^e(last)$, the optimized forwarding sequence for EED is:

$$S_{opt}^e(last,k) =$$

$$\begin{cases} S_{opt}^e(last,k-1) & EED_e(S_{opt}^e(last,k-1)) < EED_e(s_j^e \oplus S_{opt}^e(last,k-1)) \\ s_j^e \oplus S_{opt}^e(last,k-1) & Otherwise \end{cases} \tag{3.9}$$

After having all $S_{opt}^e(last,last)$ where $last \in \{1,2,\ldots,m\}$, we chose the forwarding sequence with the minimal EED value, under the constraint that $EDR \geq R$. The complexity for optimizing EED is $\mathcal{O}(Tm^2)$.

Reducing Expected Energy Consumption (EEC)

For applications such as scientific exploration, the difficulty of entering the sensing field and the corresponding high cost of system deployment calls for the longevity of the system, making energy conservation the highest priority for the system design. Similarly to the optimization of EED, if we do not have a bound on the expected delivery ratio, the optimal forwarding sequence for the minimal EEC would include only one node with the smallest EEC value in S_m^e and may also experience an extremely low source-to-sink data delivery ratio. Therefore, in this section we reduce EEC under the constraint that EDR of the forwarding sequence is above threshold R.

Unlike optimizing EED, in Equation 3.5, where i represents the index of forwarding node in the forwarding sequence, the i value changes for each already selected forwarding node as we backwardly add early nodes. In other words, the decisions made to include or exclude an **early** node in the forwarding sequence **does affect** the expected energy of **later** nodes. Lacking an optimal substructure, we can only choose either an exhaustive search (in the case that a forwarding sequence is small) or a greedy heuristic algorithm. We found that the greedy case for EEC is actually very effective. The main idea of the greedy algorithm is that starting with an empty optimal forwarding sequence, we continuously add the unselected node in S_m^e that results in a minimal increase in EEC into the optimal forwarding sequence until the EDR of the optimal forwarding sequence reaches R. Empirical results indicate that the greedy algorithm obtains optimal results 85% of the time and the suboptimal results are within 5% of the optimal values.

The Impact of EDR Constraints on Optimality

We note that the EDR bound R imposes a non-convex constraint on the EED and EEC optimization problems. To optimize the forwarding sequence efficiently, the optimization processes described in Sections 3.5.3 and 3.5.3 first identify an optimal forwarding sequence under unconstrained search space. If the resulting sequence satisfies the EDR bound R, it is also an optimal solution to the original constrained problem. However, it is also possible that the resulting sequence violates the constraint especially when the EDR bound R is very high. In this case, we select the optimal EDR forwarding sequence from S_i^e, where i is the minimal value leads to $EDR_e(S^e) \geq R$ to satisfy the constraints (instead of achieving optimal EED or EEC).

Obviously, if the percentage of constraint violation is high, our solution is not effective. To evaluate this issue, we studied the impact of a high EDR bound on the optimality of our solution. Figure 3.10 shows the percentage of optimality under different EDR bounds. Clearly, our solution is very effective in identifying optimal solutions. For example, even with a 99% delivery ratio, 98.4% solutions are optimal.

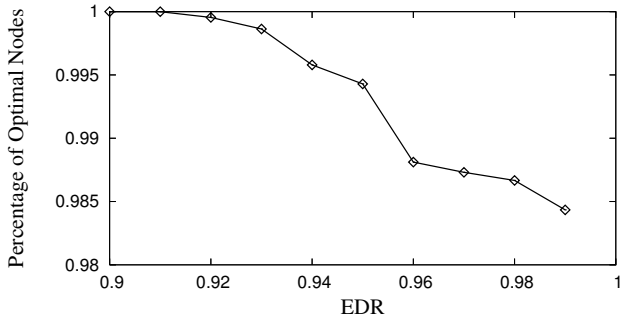


Figure 3.10: Percentage of Optimality vs. EDR

Special Cases: ETX and DESS

We note that when nodes in the network are always active with no sleeping schedules, our EDR, EED, and EEC metrics and corresponding forwarding sequences default into those of the ETX solution. In addition, when all radio links among neighboring nodes are perfect, EDR, EED, and EEC default into those in the DESS solution. To a certain degree, we argue that EDR, EED, and EEC metrics are more generic data forwarding

metrics, considering both link quality and sleep latency. In other words, ETX and DESS are two special cases of a more generic DSF solution. To validate this empirically, we will show such a convergence in the evaluation section later.

3.5.4 Distributed Implementation of DSF

In Sections 3.5.2 and 3.5.3, we discussed the model for computing EDR, EED, and EEC, and the algorithms for optimizing the forwarding sequence at a node in terms of one of those metrics. In this section, we will describe the detailed protocol implementation at each individual sensor device to compute its expectation values and decide the forwarding sequence upon the optimizing metric.

Algorithm 2 Complete protocol implementation at a node e

Input: Received expectation values from a neighboring node
Input: The current neighbor table NT of the node e
Output: The current optimal forwarding sequence
1: Update the received expectation values in NT
2: **for** Each active state in node e's working schedule ω_S **do**
3: Run either EDR, EED or EEC optimization process to obtain the optimal forwarding sequence in terms of the optimizing metric
4: **end for**
5: **if** Any node e's optimizing metric changes noticeably **then**
6: Broadcast updated expectation values
7: **end if**

As mentioned in Section 3.5.2, both the EED and EEC values of the sink node are zero while the EDR value is one, since no further in-network communication is necessary once a packet has arrived at the sink node. Therefore, at the initial time of sensor network deployment, the sink node has already had its expectation values known and is the only node that has its optimized metric value ready. The distributed algorithm for each node other than the sink node is shown in Algorithm 3.5.4. Similarly to the distributed Bellman-Ford algorithm (but using only sinks as destinations), the initialization phase starts with the sink node broadcasting its EDR, EED, and EEC values. The nodes receive the broadcasted message [Line 1], start to calculate their own expectation values according to the model and optimization process for a particular metric as discussed in the previous two sections [Lines 2-4], and then broadcast their

own expectation values if the change exceeds a certain value. This process [Lines 1-7] continues at a node until it receives no information about updated expectation values from all its neighbors. At this time, its optimizing metric also converges to its minimum at every node in the network. The convergence speed of Algorithm 3.5.4 is very fast. For example, all nodes converge within 2 seconds in our test-bed, as shown in the evaluation section.

3.5.5 Design Issues and Optimization

This section completes the description of our design by examining several design issues concerning DSF under CSMA networks, forwarding sequence optimization with simultaneous wake-up neighboring nodes, opportunistic looping, batch transmission and changing link qualities.

DSF under CSMA Networks

For the sake of clarity, we here introduce DSF in a synchronized mode. Clearly, the operations of DSF depend on neither time slots nor global time synchronization. The sufficient condition for DSF is that every node knows the *wake-up time* and the *link quality* of neighboring forwarders. To understand neighbors' wake-up times, local synchronization is needed, which can be achieved using a MAC-layer time-stamping technique [29], which achieves 2.24μs accuracy with an overhead of a few bytes of packets exchange among neighboring nodes for every 5 minutes. Since the expected τ value is set around 2000μs to 20,000μs (according to the data rates of different radio chips), an accuracy of 2.24μs is by far sufficient.

Unlike TDMA networks, DSF does not require a node to start a transmission at the beginning of a time slot τ. As long as a node knows the wake-up time of its neighboring nodes, it can decide its optimal forwarding sequence. In terms of performance, CSMA is more favorable in low duty-cycle networks where network traffic is very low, since a node in TDMA networks has to wait for its turn to transmit and becomes inefficient in such scenarios. Although DSF works in both TDMA and CSMA networks, CSMA is a better choice.

Simultaneous Wake-Up

In Section 3.5.3, for the sake of simplicity we optimized the forwarding sequence under the assumption that at each node, no multiple neighboring nodes would wake up at the same time interval. In this section, we complete the forwarding sequence optimization process to deal with multiple neighbors waking up at the same time.

In contrast to Section 3.5.3, when we attempt to include a node j from S^e_m into $S^e_{opt}(k-1)$, instead of only inserting the node j to the front of $S^e_{opt}(k-1)$, there are

two possible actions:

- If the wake-up time associated with node j is different from the wake-up time associated with the first node in sequence $S^e_{opt}(k-1)$, we just append node j to the front of $S^e_{opt}(k-1)$ as described in Section 3.5.3.
- If the wake-up time associated with node j is the same as the wake-up time associated with the first node in sequence $S^e_{opt}(k-1)$, we ***replace*** the first node in $S^e_{opt}(k-1)$ with the node j.

After properly including the node j into sequence $S^e_{opt}(k-1)$, we can follow the same procedure presented in Section 3.5.3, comparing the optimizing metric for new sequence with $S^e_{opt}(k-1)$ and deciding if node j should be within $S^e_{opt}(k)$ or not.

Opportunistic Looping

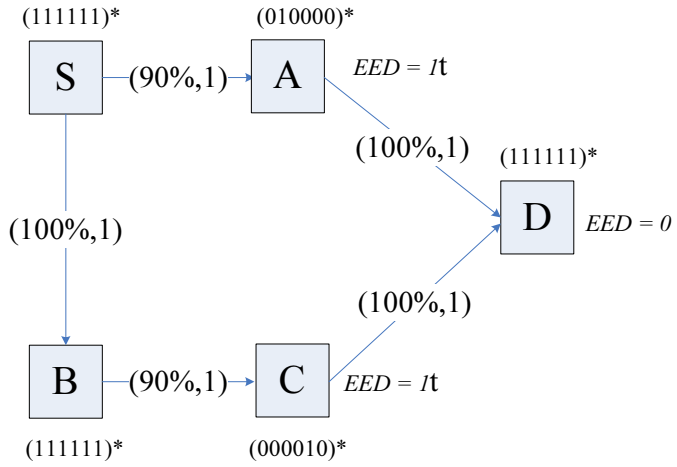


Figure 3.11: Opportunistic Looping

We note that the forwarding sequence optimization for EED (Section 3.5.3) does not require that a potential forwarder has a smaller EED value than the sender (i.e., a forwarder could be chosen even if it has a large EED value as long as the participation of this forwarder can reduce the sender's EED value). This relaxation allows a packet to travel through temporary loops in the path to its destination to potentially reduce the end-to-end delay! We term this interesting, albeit counterintuitive, phenomena

opportunistic looping. Figure 3.11 shows an example of opportunistic looping, in which node S sends a packet to node D. Each node is assigned a working schedule and each edge is assigned a tuple (p, d), in which p is round-trip success ratio p and d is waiting time. We also tag each node with an EED value, calculated by Equation 3.4.

As shown in Figure 3.11, node S can deliver a packet to node A with a 90% success ratio. If successful, the packet arrives at node A and then arrives at node D with total delay of two. In case of a delivery failure between S and A, node S has two options. In the first option, node S does nothing, but waits for five units of time before it tries A again. The other option is to forward this packet to node B in one unit of time, and then node B tries to deliver the packet to node D through node C. In the later case, even if node B fails to deliver the packet to node C, node B can loop this packet back to S before node A becomes available again. Obviously, the loop $S \rightarrow B \rightarrow S$ is opportunistic in nature. In other words, compared with idle waiting, this loop can potentially reduce delay if transmission from node B to node C is successful.

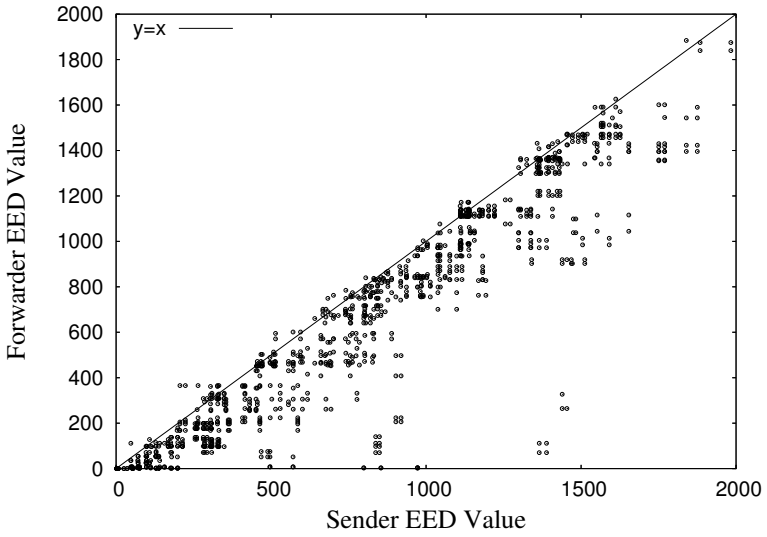


Figure 3.12: Forwarder EED Values vs. Sender EED Values

In the results of our simulation study, we find that the behavior of selecting forwarders with larger EED values is not rare. Figure 3.12 shows the forwarders' EED values versus the senders' EED values in a random 2000-node network. There we can see that a fairly large number of nodes have their forwarders with larger EED values.

Based on simulation results from 100 randomly generated networks, we observe that nearly one-tenth of the nodes have forwarders with larger EED values.

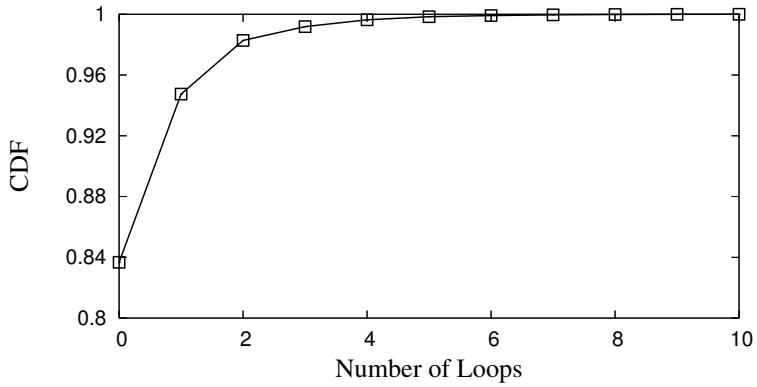


Figure 3.13: Number of Loops in E2E Paths

We also investigated opportunistic looping from the perspective of the E2E path. The CDF curve in Figure 3.13 shows the probability that a packet traveling through multiple loops decays exponentially. For example, 84% of packets are delivered without loops, while only 0.4% of packets are delivered through 4 hops.

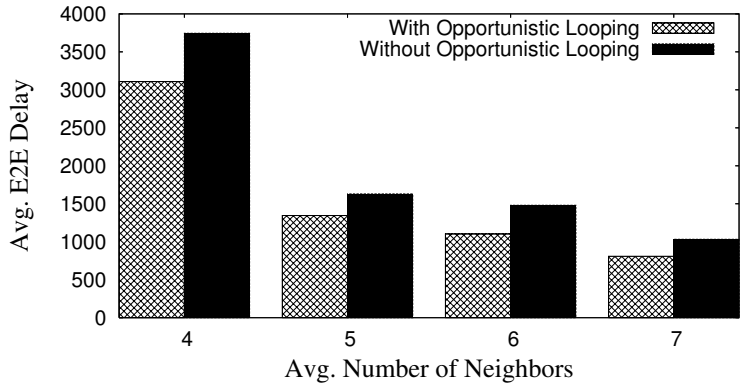


Figure 3.14: Delay with and without Opportunistic Looping

Opportunistic looping can be optionally disabled by requiring that forwarders have a smaller EED value during the forwarding selection process. Figure 3.14 compares the selection process with and without opportunistic looping. Clearly, opportunistic looping can noticeably reduce the end-to-end delay. For example, with an average

of 4 neighboring nodes, the E2E delay for with and without loops is 3109 and 3745, respectively.

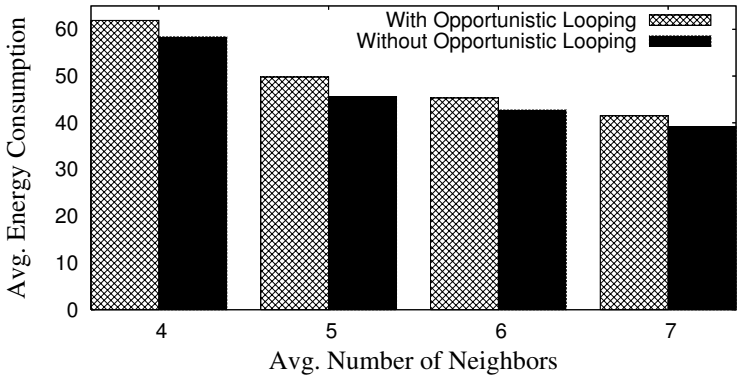


Figure 3.15: Energy with and without Opportunistic Looping

On the other hand, it should be emphasized that although opportunistic looping reduces end-to-end delay, it introduces additional communication overhead. Figure 3.15 compares the average EEC with and without opportunistic looping. By comparing Figures 3.14 and 3.15, a system designer can decide whether to use opportunistic looping based on the tradeoffs between the delay requirement and the energy budget.

Batch Transmission

While describing the network model in Section 3.4, we assume that during time τ, at most one packet can be transmitted between a sender and a receiver. This is true if sensors are equipped with slow radio chips. For sensor nodes with fast radio (e.g., MicaZ with $250kbps$ CC2420 radio [57]), however, it is possible to transmit multiple packets with time τ. In addition, with a relatively large τ value (e.g.,100 milliseconds), the accuracy requirement on time synchronization (in coverage scheduling) can be reduced, since such state-of-the-art solutions as FTSP [29] can easily achieve sub-millisecond accuracy.

Besides providing relief in the time synchronization constraint, the relatively large τ also helps reduce the E2E delay of the source-to-sink communication. Instead of attempting to transmit only one packet during one τ, as discussed in Section 3.5, the sending node now could take advantage of a longer duration of τ and repeatedly transmit the packet until the packet has been successfully received or the duration of τ ends.

We need only a simple modification of the basic design to support batch transmission. Specifically, we rewrite p_{ei}, the bi-directional link quality between two nodes in Section 3.5.2, as $p'_{ei} = 1 - (1 - p_{ei})^m$, where m is the maximum number of retransmissions allowed during one τ. Essentially, the new p'_{ei} value represents the probability that the receiver received the packet by m transmissions.

We also note that although the increasing τ improves the packet delivery rate over that of one link, the increased τ also adds more delay to the network. Therefore, a suitable m and consequently τ could be derived to further minimize the source-to-sink delay.

Accommodating Link Quality Change

Clearly link quality could change over a long period of time [58], and therefore the DSF design must be able to accommodate such changes. As described in Section 3.5.4, the implementation of DSF allows each node to re-evaluate its own EDR, EED, and EEC values with updated information from neighboring nodes. The continuous updates might not be a suitable solution if energy consumption is the paramount concern. To reduce the energy consumption and avoid oscillation in forwarding decisions, we permit information exchange only when changes in link quality exceed a certain threshold. This would lead to temporary suboptimal metrics, since it is possible that the outdated link quality information could be used for optimization.

3.6 Implementation and Evaluation

We have implemented a complete version of the DSF forwarding scheme on the TinyOS/Mote platform in nesC [59] with 20 MicaZ motes. To compare performance, we also implemented ETX [35] on the motes. The major components of DSF implementation include neighbor discovery, link quality measurement, the forwarding sequence optimization algorithms discussed in Section 3.5.3, and data forwarding with an optimized forwarding sequence.

- **Neighbor Discovery:** The neighbor discovery component at each individual node manages both the broadcasting of the working schedule and the maintenance of the neighbor table. To announce the existence of a node, the neighbor discovery component broadcasts the node ID and working schedule information with a configurable parameter that decides the number of retransmissions. In current prototype implementation, we set the number of retransmissions 10. While receiving a broadcasted working schedule announcement, the neighbor discovery component checks whether the source of the packet has been in its neighbor table. If the source does not exist in the neighbor table, the neighbor discovery component appends the source and corresponding working schedule into its neighbor table. Otherwise, the neighbor discovery component would just ignore the received broadcasting packet and does nothing. In short, the neighbor discovery component attempts to keep track of the schedules of all neighbors.

- **Link Quality Measurement:** To measure the pairwise link quality between a node and its neighbors, the link quality measurement component at each individual node sends a number of packets to each of its neighbors and utilizes the link layer acknowledgement from B-MAC [49] to calculate the pairwise link quality. Depending on the desired accuracy of measurement, the link quality measurement component provides a configurable parameter to set the number of message transmissions between pairs of nodes. In order to minimize the impact of interference, in our current implementation we serialize the link quality measurement process among nodes in the network; in other words, nodes in the network measure their

link qualities in sequence. When the data forwarding component forwards packets, the link quality measurement component updates link quality information accordingly and triggers forwarding sequence optimization if necessary.

- **Forwarding Sequence Optimization:** Currently the heart of DSF design, the forwarding sequence optimization component implements EDR and EED optimizations faithfully according to the description in Section 3.5.3. The two optimizations are necessary to create optimal forwarding sequences of EED for comparison with ETX in the following subsection. In the future, we also plan to complete the forwarding sequence optimization component with inclusion of EEC implementation.

- **Data Forwarding:** The data forwarding component is shared by both DSF and ETX. Whenever a node has a packet ready to send, according to the specified forwarding scheme, the data forwarding component at a node attempts a single packet transmission to the designated forwarding node when it is in the active state.

We use FTSP [29] for the purpose of time synchronization among motes and Deluge [20] for the purpose of wireless reprogramming. The compiled image occupies 27,398 bytes of code memory and 1,137 bytes of data memory.

During the experiment, we *randomly* placed 20 MicaZ motes along the hallway of our office building and tuned the transmission power to ensure the multi-hop communication between the source node and the sink node. In this experiment, immediately after deployment, all nodes are in the initialization phase, with all nodes keep awake. Each node randomly generates a periodic 1% working schedule, represented as a regular binary string as described in Section 3.4 with switching rate τ sets to be $20ms$ (a relatively large τ value here to reduce the impact of the time synchronization module). After generating working schedules, nodes start to broadcast their own working schedules and measure the pairwise link quality for their neighboring nodes. With known working schedules of neighboring nodes and corresponding link qualities, nodes in the network start to carry out the Algorithm 3.5.4, calculate the specified metrics and decide the forwarding sequence for DSF. ETX shares the identical information as DSF and builds its own forwarding metrics accordingly. After Algorithm 3.5.4 and ETX converge at

a node, the node begins to execute its working schedule with a timer-driven FA logic, turns off the radio at dormant bits, and enables the radio at the active bits. At the data forwarding phase, the node furthest away from the sink is selected as the source node and sends the packets using DSF and ETX alternatively so as to minimize the impact of temporal link qualities.

This testbed experiment was repeated multiple times with different node placement and working schedules. The results show the similar trend that resulted in all the experiments, and we report one collected dataset from the experiments in the following subsection.

3.6.1 Performance Comparison

In this section, we describe and compare the empirical E2E delay and energy consumption for DSF and ETX. In the experiment, the source node sends 100 packets to the sink node with DSF of optimal EED and ETX forwarding scheme, respectively.

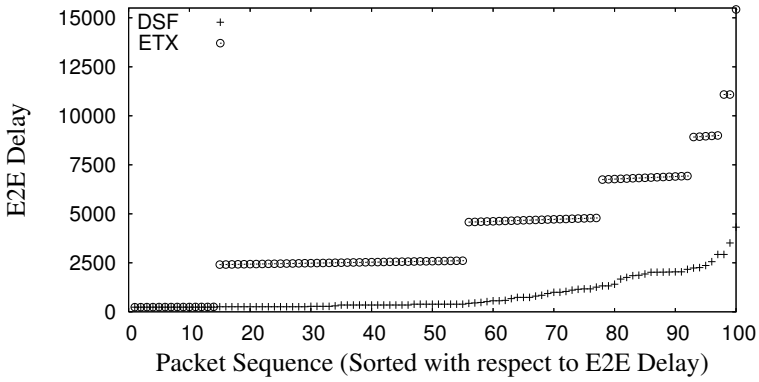


Figure 3.16: E2E Data Delivery Delay

Figure 3.16 shows the E2E data delivery delay for DSF and ETX. The packets in the figure are sorted according to their E2E delay, making it clear that ETX experiences heavy penalties when its single-hop transmission has failed, since it has to wait for the fixed forwarding node to wake up again. In contrast, when DSF encounters a single-hop transmission failure, its capability to dynamically switch the forwarding node significantly reduces the E2E delay. For instance, among 100 sent packets, the maximal

E2E data delivery delays for DSF and ETX are 4317ms and 15426ms respectively, while the average delays are 849ms and 3942ms.

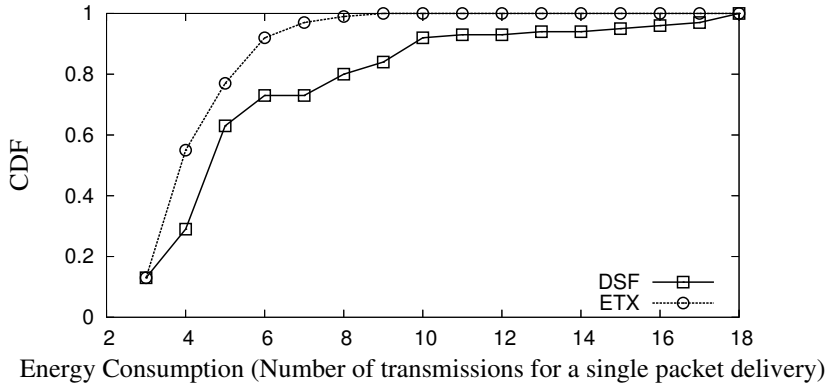


Figure 3.17: Energy Consumption

In addition to the E2E delay, we are also interested in the energy consumption of the two comparing protocols. Figure 3.17 demonstrates the energy consumption (number of transmissions for a single packet delivery) for DSF and ETX. From the figure, we can see that ETX incurs a smaller number of transmissions than DSF. For example, all of the packet deliveries for ETX finished with a maximum of 9 transmissions, while about 84% of the packets for DSF arrived at the sink node within 9 transmissions. However, the DSF shows a better delay-energy efficiency than ETX. With the same 9 transmissions, the delay for DSF and ETX is 1785ms and 15426ms, respectively.

3.6.2 System Insights

In this section, we investigate the internal state of each sensor node and reveal the corresponding statistics for DSF.

Figure 3.18 demonstrates the greater diversity of forwarder link qualities for DSF over those for ETX. While almost all ETX forwarders have link qualities above 50%, the distribution of forwarder link qualities for DSF is roughly uniform and ranges from 3% to 97%. Such diversity in forwarder link qualities for DSF, along with its smaller E2E delay, leads us to conclude that unreliable links are also helpful in reducing E2E delays in low duty-cycle sensor networks.

Figure 3.19 shows the relationship between the number of nodes in the forwarding sequence and the number of available neighboring nodes for each sensor node in the

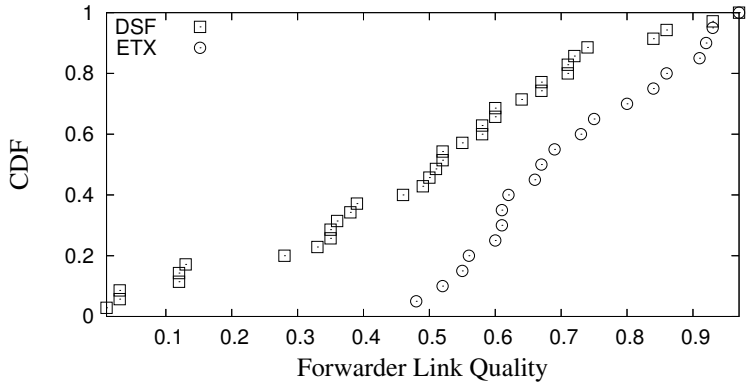


Figure 3.18: Diversity in Forwarder Link Qualities

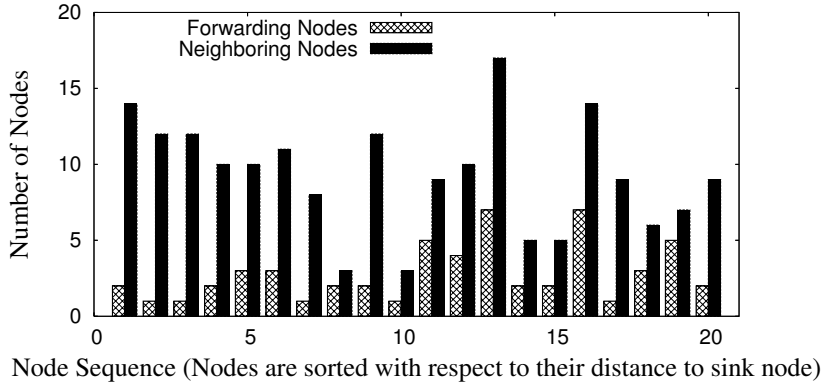


Figure 3.19: Number of Forwarding Nodes vs. Number of Neighboring Nodes

experiment. The node sequence is ordered by the node's distance to the sink node. From this figure, we can see that most nodes have more than one node in their forwarding sequence. We also observe that generally, as the node's distance to the sink node increases, the number of forwarding nodes in the forwarding sequence also increases, since in order to maintain a certain data delivery ratio, the more distant nodes normally need to select more of their neighboring nodes. For example, the average number of forwarding nodes for the first 10 nodes is 1.8 nodes, while the value for the last 10 nodes is 3.8 nodes.

In addition to studying the distribution of the forwarding nodes, we also investigated how fast each node converges to its optimal forwarding sequence. To track the convergence speed of the DSF, we recorded the number of times that each node executed its

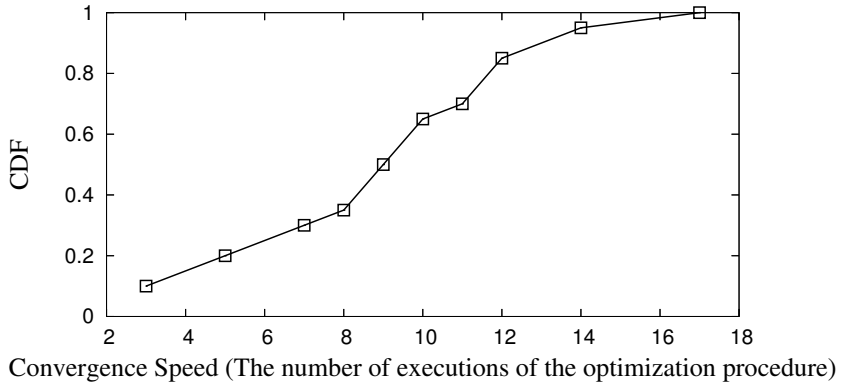


Figure 3.20: DSF Convergence Speed

forwarding sequence optimization procedure, as shown in Figure 3.20. There we can see that the forwarding sequence optimization process at all nodes converges within 18 executions of the optimization procedure. Furthermore, the number of executions of the optimization procedure at individual nodes is proportional to the number of neighboring nodes. This observation is also consistent with our complexity analysis for forwarding sequence optimization procedures.

3.7 Large-Scale Simulation

The results of the following system evaluation indicates that our proposed approaches can be efficiently implemented on resource-constrained sensor nodes and demonstrates their effectiveness in improving source-to-sink wireless communication between sensing nodes and sink. However, this evaluation was restricted to a limited design space. In order to understand the performance of the proposed scheme under numerous network settings, in this section, we provide simulation results with 250 nodes. We compared the performance of DSF with following state-of-the-art solutions:

- ETX [35] by Douglas S. J. De Couto et al. in Mobicom'03.
- PRR×D [36] by Karim Seada et al. in SenSys'04.
- DESS [37] by Gang Lu et al. in INFOCOM'05.

3.7.1 Simulation Setup

In the simulation, we deployed 250 sensor nodes randomly in a 150m×150m square field. A sink was positioned in the center of the deployment field, and each sensor node sent its packet to the sink over multiple hops. The radio model was implemented according to [53], which considers the oscillation nature of the radio links and has several adjustable parameters. Except as otherwise specified, we set these parameters strictly according to the CC2420 radio hardware specification [57]. These parameters accurately reflect the performance of MicaZ motes in that they have the same modulation method, encoding method, frame length and path loss exponent.

In all experiments, we set the sender retransmission time bound T equals 200τ, which is also the length of the node working schedule. Each experiment was repeated 30 times with different random seeds, node deployments, and node working schedules. Data collected at each node was obtained by averaging 1000 source-to-sink communications. The 95% confidence intervals are within 1∼10% of the means.

3.7.2 Performance Evaluation

This section compares the data delivery ratio, E2E delay and energy consumption per delivered packet of source-to-sink communications among DSF, ETX, PRR×D, and DESS under different link qualities and duty cycles.

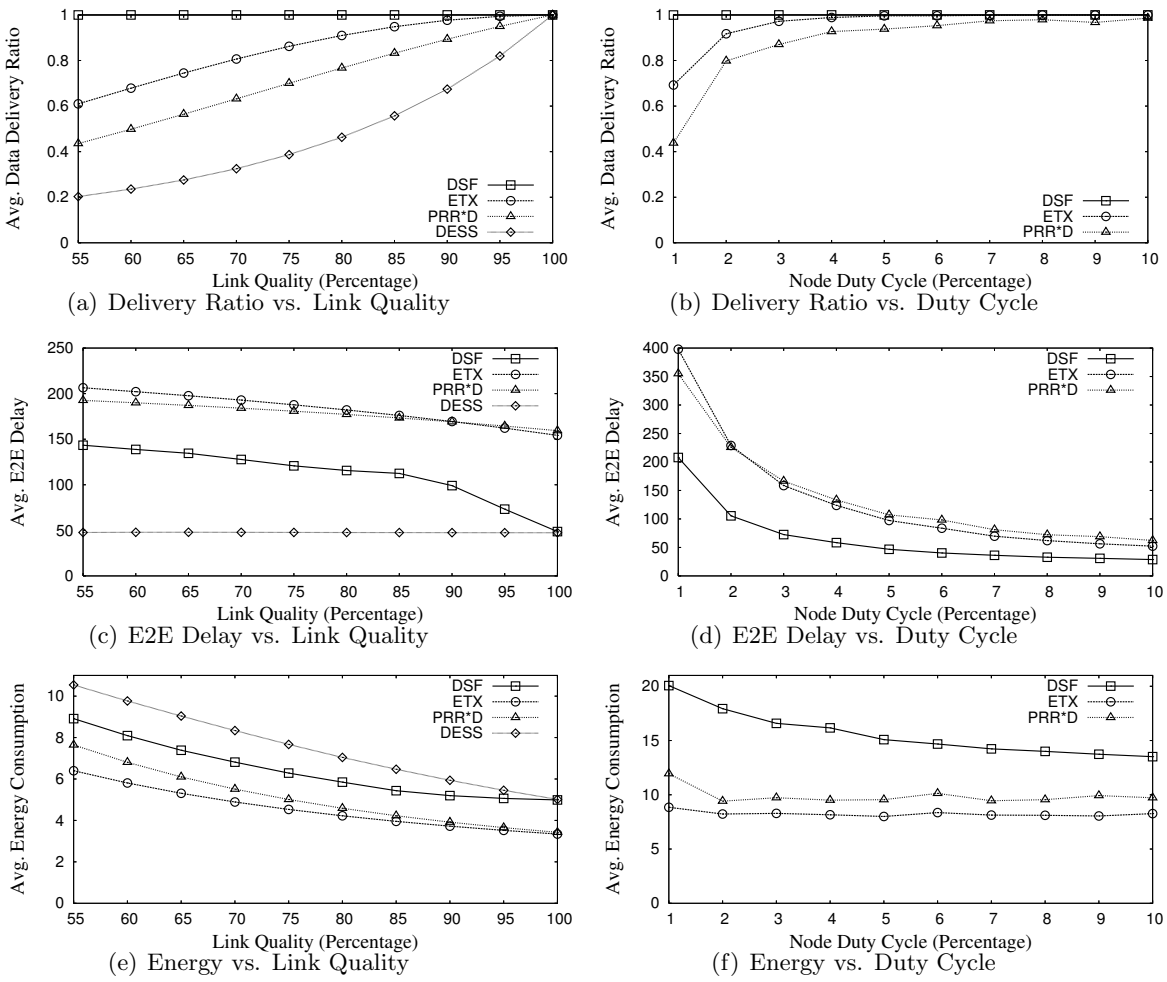
For the simulation of different link qualities, we first used CC2420 radio specifications to obtain the neighbor table for each sensor node, then set the pairwise link quality according to the simulation configurations.

In following three subsections, evaluation figures for optimizing metrics are shaded to highlight their performances.

Optimizing Expected Delivery Ratio

In this section, we examine the performance difference among DSF with optimal EDR, ETX, PRR×D, and DESS under different link qualities and duty cycles.

Varying Link Qualities: Figure 3.21(a) shows the data delivery ratio among the four compared schemes. The figure clearly shows that under the low link qualities, ETX, PRR×D and DESS can deliver only a very small portion of packets, while DSF with



(a) Delivery Ratio vs. Link Quality
(b) Delivery Ratio vs. Duty Cycle
(c) E2E Delay vs. Link Quality
(d) E2E Delay vs. Duty Cycle
(e) Energy vs. Link Quality
(f) Energy vs. Duty Cycle

Figure 3.21: Optimizing Expected Delivery Ratio (EDR)

optimal EDR is able to deliver most of the packets to the sink node. For example, when the link quality is 55%, DSF delivers 99.9% of packets, while ETX, PRR×D, and DESS deliver only 61.0%, 43.5% and 20.3% of packets, respectively. Therefore, when the data delivery ratio is the primary design goal of a sensornet application, DSF would be a good choice for the system.

Figure 3.21(c) and Figure 3.21(e) show the corresponding E2E delay and energy consumption for four schemes. From Figure 3.21(c), we observe that DESS has the smallest and most constant E2E delay at all link qualities because at each hop, DESS

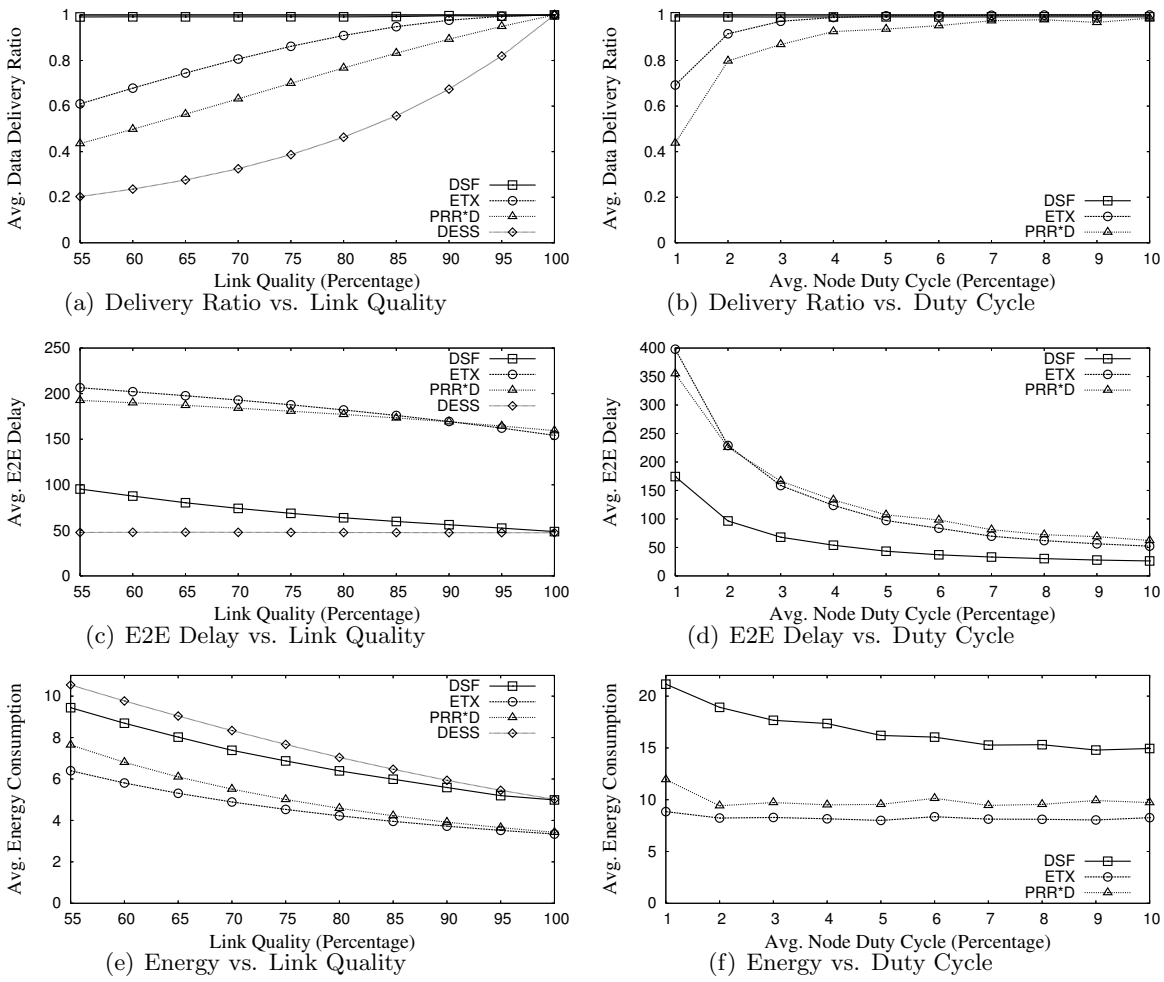
would attempt to transmit its packet to the forwarder only once on the shortest delay path during one round of the node working schedule. Therefore, all the packets for DESS that reach the sink node are those for which every single-hop transmission is successful with one single attempt, and that consequently represent the minimal possible delivery delay, which is a constant value. At the same time, however, DESS experiences the largest packet loss among the four compared schemes. DSF, on the contrary, has the largest data delivery ratio though a smaller E2E delay than ETX and PRR×D. However, DSF's high data delivery ratio also incurs energy penalties.

From Figure 3.21(e), we can see that DSF has a slightly higher energy consumption per delivered packet than ETX and PRR×D since it attempts more transmissions and delivers more packets than these schemes. DESS ignores the link quality completely, has a very low data delivery ratio, and wastes much energy on transmitting packets that do not arrive at the sink node, therefore having the largest energy consumption per delivered packet. For instance, at a link quality of 55%, the per-delivered packet energy consumption for DSF, ETX, PRR×D, and DESS is 8.91, 6.40, 7.64 and 10.54, respectively.

Varying Duty Cycles: Figure 3.21(b) reports the data delivery ratio under different node duty cycles. It shows that under all node duty cycles, DSF with optimal EDR has a higher data delivery ratio than ETX and PRR×D. As the node duty cycle increases, the data delivery ratio for all schemes increases as well. For example, the delivery ratio for DSF, ETX, and PRR×D increases from 99.9%, 69.3%, and 43.8% to 100%, 99.9%, and 98.6%, respectively, when duty cycle increases from 1% to 10%. Figure 3.21(d) shows that the corresponding E2E delay for DSF is smaller than the other two baseline schemes, even with a higher data delivery ratio. Figure 3.21(f) shows again that the high data delivery ratio of DSF results in higher energy consumption.

Optimizing Expected E2E Delay

In this section, we examine the performance difference among DSF with optimal EED, ETX, PRR×D, and DESS under different link qualities and duty cycles. For optimal EED at each node, we set the data delivery ratio bound as 99%.

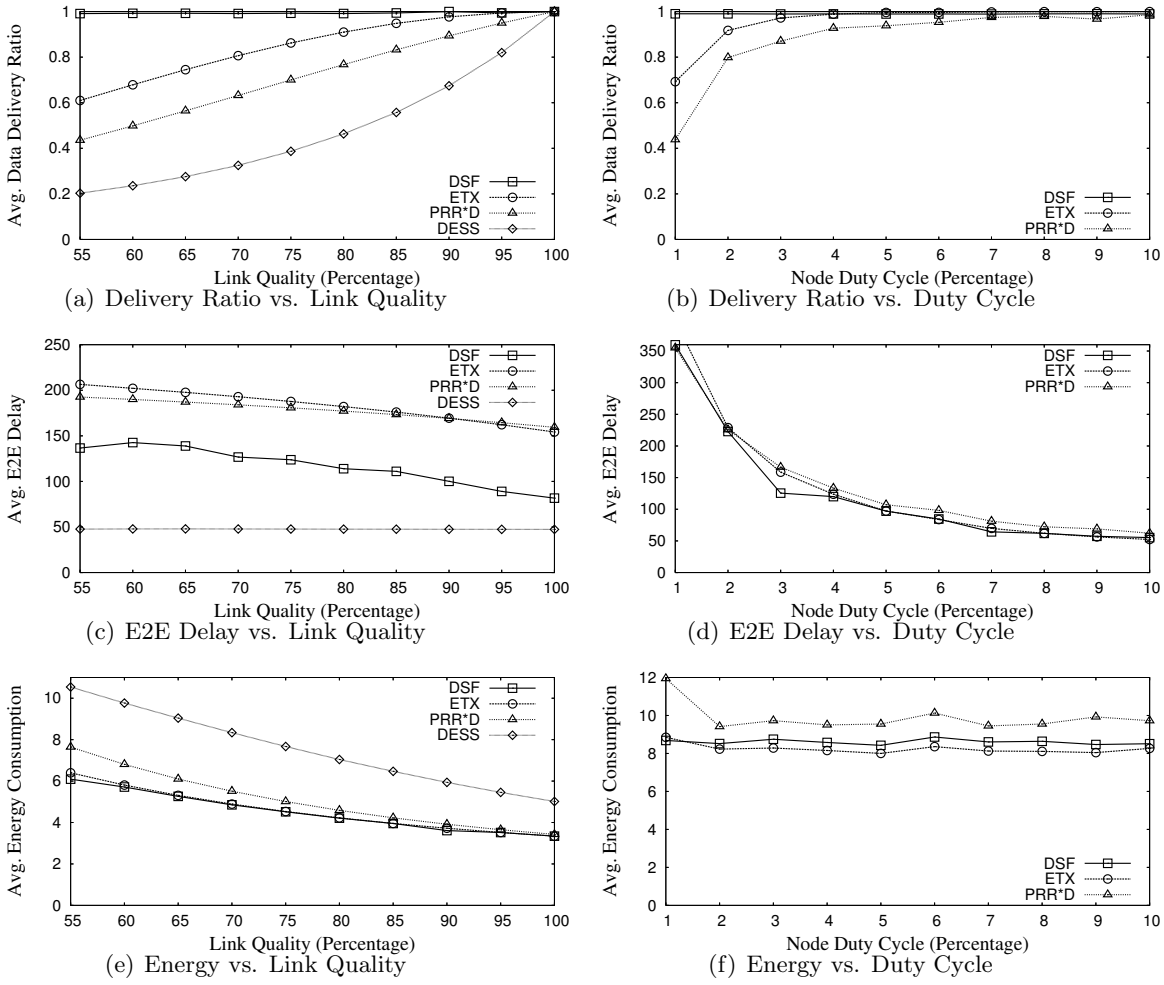


(a) Delivery Ratio vs. Link Quality
(b) Delivery Ratio vs. Duty Cycle
(c) E2E Delay vs. Link Quality
(d) E2E Delay vs. Duty Cycle
(e) Energy vs. Link Quality
(f) Energy vs. Duty Cycle

Figure 3.22: Optimizing Expected E2E Delay (EED)

Varying Link Qualities: Figure 3.22(c) shows the end-to-end delay for four forwarding schemes under different link qualities. At link qualities less than 100%, the E2E delay is larger for DSF than for DESS, for the reason mentioned in the previous subsection. Meanwhile, the E2E delay for DSF is much smaller than for ETX and PRR×D. For example, at a link quality of 90%, the E2E delay for DSF, ETX, and PRR×D is 56.2, 169.4, and 178.3, respectively. When the link quality reaches 100%, the results for DSF with optimal EED converges with those of DESS. In Figure 3.22(e), we can see that the energy consumption for DSF is still higher than that for ETX and PRR×D. However,

we also observe that DSF is more delay-energy efficient than the other schemes. For example, when the link quality is 80%, the per-energy delay for DSF, ETX, PRR×D, and DESS is 10.07, 47.41, 50.36, and 14.61, respectively.



(a) Delivery Ratio vs. Link Quality
(b) Delivery Ratio vs. Duty Cycle
(c) E2E Delay vs. Link Quality
(d) E2E Delay vs. Duty Cycle
(e) Energy vs. Link Quality
(f) Energy vs. Duty Cycle

Figure 3.23: Reducing Expected Energy Consumption (EEC)

Varying Duty Cycles: Figure 3.22(d) shows the end-to-end communication delay under different node duty cycles. There we can see that DSF has a smaller delay than the baseline schemes under all duty cycles while retaining a high data delivery ratio (Figure 3.22(b)). The overall energy consumption for DSF is still higher than that for the other schemes. However, as mentioned before, the per-energy delay for DSF is much

smaller than for ETX and PRR×D. For example, at a duty cycle of 5%, the per-energy delay for DSF, ETX, and PRR×D is 3.82, 14.08, and 16.33, respectively.

Reducing Expected Energy Consumption

This section presents the performance differences among DSF with optimal EEC, ETX, PRR×D, and DESS under different link qualities and duty cycles. For an optimal EEC at each node, we set the data delivery ratio bound as 99%.

Varying Link Qualities: In Figure 3.23(e), energy consumption for DSF approaches the ETX at all link qualities while maintaining high data delivery ratio. For example, when link quality is 70%, the energy consumption for DSF, ETX, PRR×D, and DESS is 4.21, 4.21, 4.59, and 7.04, respectively. When link quality approaches 100%, DSF converges to the ETX in terms of energy consumption. In addition, with equivalent energy consumption, the E2E delay for DSF is smaller than for ETX and PRR×D. At a link quality of 80%, the E2E delay for DSF, ETX, and PRR×D is 113.95, 182.13, and 197.18, respectively. Interestingly, we notice that under optimal EEC, DSF does not converge to the DESS when link quality reaches 100%, because when optimizing EEC, DSF would seek the delivery path with the minimum number of transmissions instead of the minimum E2E Delay.

Varying Duty Cycles: Figure 3.23(f) shows the energy consumption under different node duty cycles. From the figure, we observe that the energy consumption for DSF approaches that of ETX and is better than that of PRR×D. With a higher data delivery ratio (Figure 3.23(b)) and comparable energy consumption, the end-to-end delay for DSF is still smaller than for the baseline schemes.

3.7.3 Insights

In the previous section, we saw the significant improvement of the source-to-sink communication for DSF over ETX, PRR×D, and DESS. In this section, we reveal the underlying reasons why DSF provides better performance than those state-of-the-art solutions.

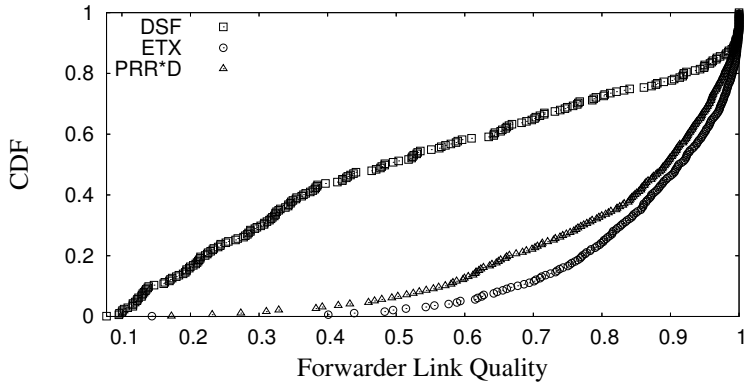


Figure 3.24: Diversity in Forwarder Link Qualities (Average # of Neighbors=6)

Diversity in Link Quality

Both ETX and PRR×D generally prefer reliable links and try to avoid highly unstable links. While this intuitive approach holds well in traditional wireless networks, we saw that as node duty cycle decreases, the delay of such schemes becomes excessive since the time spent on waiting for the forwarder to wake up again is no longer tolerable. Figure 3.24 shows the CDF curve of the forwarder's link qualities for 200 randomly sampled senders from DSF, ETX, and PRR×D. From the figure, we can see that the distribution of DSF link quality is roughly uniform, with no obvious range being favored, while ETX and PRR×D select much more reliable links. This observation strengthens our understanding that unreliable links are as useful as highly reliable links for minimizing the source-to-sink communication delay in low duty-cycle networks.

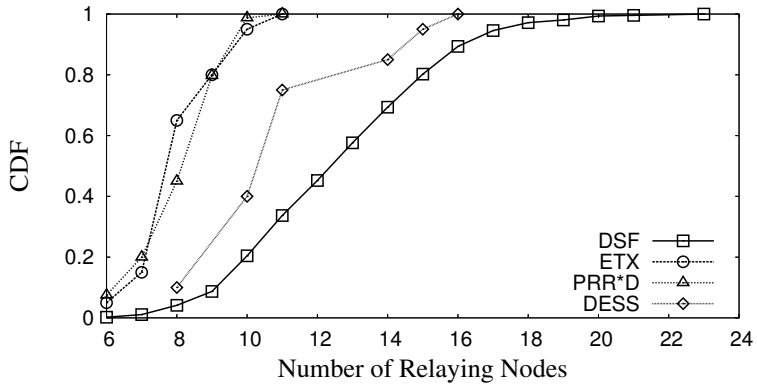


Figure 3.25: Diversity in Delivery Paths (# of Neighbors=6)

Diversity in Delivery Paths

In the previous subsection, we demonstrated that picking low-quality links is beneficial in low duty-cycle sensor networks for reducing the source-to-sink communication delay. In this section, we show the greater diversity of delivery paths for DSF over those for ETX, PRR×D, and DESS. In the simulation setup, 150 nodes are deployed in a $160m \times 160m$ field. Forty source nodes on the edge of the field send their packets to the sink node located in the center of the field. In Figure 3.25, we show the number of nodes that relay the packets sent by the source nodes during 100-packet delivery processes for DSF, ETX, PRR×D, and DESS. Clearly, DSF explores a much larger neighbor space than the other three schemes in these 100 packet transmission processes. For example, the maximum number of relaying nodes for DSF, ETX, PRR×D, and DESS is 23, 11, 11, and 16, respectively. This again demonstrates DSF's adaptability to the presence of unreliable radio links and the low duty-cycle of sensor nodes.

3.8 Conclusion

In this work, we propose a dynamic switch-based forwarding (DSF) scheme for extremely low duty-cycle sensor networks, which addresses the combined effect of unreliable radio links and sleep latency in data forwarding. We derive a distributed model for data delivery ratio (EDR), E2E delay (EED), and energy consumption (EEC) at individual nodes and optimize the forwarding action in terms of these three metrics. To evaluate the performance of DSF, we have fully implemented the DSF in a network of 20 MicaZ motes and performed extensive simulation with various network configurations. The results demonstrate that DSF significantly improves source-to-sink communication over several state-of-the-art solutions in low duty-cycle sensor networks with unreliable radio links.

Chapter 4

Energy Synchronized Communication in Energy-Harvesting Networks

With advances in energy harvesting techniques, it is now feasible to build sustainable sensor networks (SSN) to support long-term applications. Unlike battery-powered sensor networks, the objective of sustainable sensor networks is to effectively utilize a continuous stream of ambient energy. Instead of pushing the limits of energy conservation, we are aiming at energy-synchronized designs[1] to keep energy supplies and demands in balance. Specifically, this chapter presents the Energy Synchronized Communication (ESC) as a transparent middleware between the network layer and data link layer that controls the amount and timing of RF activity at receiving nodes. In this chapter, we first derive a delay model for cross-traffic at individual nodes, which reveals an interesting *stair effect* in low-duty-cycle networks. This effect allows us to design a localized energy synchronization control with $\mathcal{O}(1)$ time complexity that *shuffles* or *adjusts* the working schedule of a node to optimize cross-traffic delays in the presence of changing duty-cycle budgets. Under different rates of energy fluctuations, shuffle-based and

[1] Usually synchronization refers to time dimension, for energy-harvesting sensor networks we propose to use the term synchronization to represent the balance between energy supply and demand in the network.

adjustment-based methods have different influences on *logical connectivity* and *cross-traffic delay*, due to the inconsistent views of working schedules among neighboring nodes before schedule updates. We study the tradeoff between them and propose methods to update working schedules efficiently. To evaluate our work, ESC is implemented on MicaZ nodes with two state-of-the-art routing protocols. Both test-bed experiment and large scale simulation results show significant performance improvements over randomized synchronization controls.

4.1 Introduction

With the increasing need for cyber-physical interaction, Wireless Sensor Networks (WSN) have emerged as a key technology for many long-term applications, such as military surveillance [60, 61], field monitoring [62] and assisted living [63]. Due to the stringent constraints on cost and form factors, traditional battery-powered sensor networks must balance the tradeoff between sustainability and system performance [64, 65, 66]. Normally, the design objective of these systems is to *conserve as much energy as possible* while meeting minimal requirements [17, 67, 68].

Because sensor networks usually interact with the physical environment, they are especially well-suited to exploit ambient energy resources. For example, there are already many existing technologies to extract energy from the environment such as solar, thermal, optical, and kinetic energies [69, 70, 71]. In addition, several recent works have built prototypes [72, 73, 74] to demonstrate the feasibility of deploying *sustainable sensor network* with energy-harvesting nodes. However, influenced by the traditional belief in energy management, the designers of those systems still think that *maximum* energy harvesting and *minimum* energy consumption are always beneficial.

Different from previously ingrained belief, we take a new position in this work. We argue that energy management should *synchronize* the supply with demand. The equilibrium point is achieved when the energy supplied and demanded are in balance. It is not always beneficial to conserve energy when a network can harvest excessive energy from the environment since energy storage devices (e.g., batteries or capacitors) are always limited in capacity and usually leakage-prone. Therefore, energy saving

with reduced performance during energy-rich periods is actually wasteful and counter-productive. In other words, in sustainable sensor networks, we would like to *consume as much energy as possible* while maintaining their sustainability.

In this work, we are particularly interested in studying the impact of energy synchronization on communication performance. We propose *Energy Synchronized Communication* (ESC), a novel solution that *dynamically synchronizes node activity patterns with available energy budgets, so as to minimize communication delay at individual nodes* in sustainable sensor networks. Specifically, by exploiting an interesting stair-effect of delay during energy synchronization, ESC is capable of minimizing communication delay at individual nodes in constant time and lies as a generic middleware between the data link layer and network layer. The major intellectual contributions of this work are as follows:

- An Energy Synchronized Communication (ESC) protocol is designed as a generic middleware service for supporting existing network protocols. To our best knowledge, this is the first in-depth and generic work to study low duty-cycle communication performance in energy-harvesting, sustainable sensor networks.

- We formulate a general model for cross-traffic delay at individual nodes. This model reveals an interesting *stair effect of delays* in low duty-cycle networks. We show that, counterintuitively, the communication delay is not affected by when a packet is received as long as the packet arrives within a certain interval. This stair effect allows us to design a localized $\mathcal{O}(1)$ algorithm to minimize communication delay while synchronizing duty-cycles of individual nodes with available ambient energy.

- *Logical Link Quality (LLQ)* is put forward to reflect the true reliability of a link in low-duty-cycle networks, capturing transient inconsistency views among neighboring nodes. We study the impact of *shuffle-based* and *adjustment-based* synchronization on LLQ and propose how to maintain LLQ with a small overhead.

The rest of the chapter is organized as follows: Section 4.2 motivates the necessity of ESC. Section 4.3 articulates the network model and related assumptions. Section 4.4 formalizes a delay model for cross traffic. Section 4.5 and Section 4.6 describe our

energy-synchronized control algorithm and the impact of updates on logical link quality. Section 4.7 discusses practical design issues. Section 4.8 and Section 4.9 present testbed and simulation evaluation results, respectively. Section 4.10 briefly discusses related work. Section 4.11 concludes the chapter.

4.2 Motivation

The motivation of this work is originated from our empirical experience of deploying energy-harvesting, sustainable sensor networks. In such networks, the energy supply usually is very dynamic and the whole network normally has to operate at low-duty-cycle to ensure the sustainability. With those unique characteristics, effective data communication in such networks therefore poses a new challenge beyond traditional static, battery-powered sensor networks. In the following paragraphs, we explain the impact of dynamic energy supply and low-duty-cycle networking on data communication process in detail.

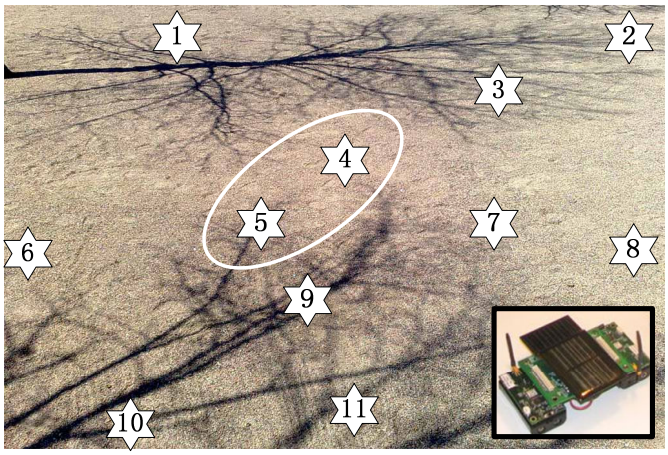
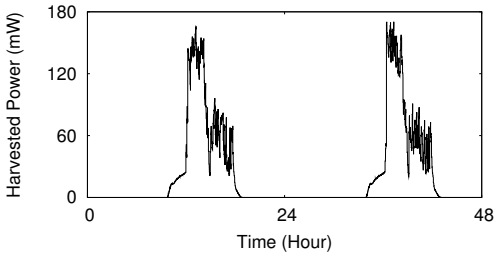
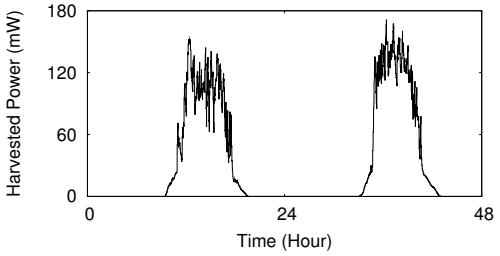


Figure 4.1: Experiment Site

First, in energy harvesting networks, the harvested energy usually is very unpredictable and could vary significantly over time [73, 74]. To study the empirical energy harvesting rate over time, we have deployed 11 solar-powered sensor nodes in an open field, shown in Figure 4.1. We collect the energy harvesting rates for the 11 deployed sensor nodes for a period of two days and Figure 4.2 plots the harvested energy over time for node 4 and node 5 in Figure 4.1. Clearly in Figure 4.2, in the time dimension, the harvested energy varies significantly both within a day and across days. Furthermore, in the space dimension, even node 4 and node 5 are physically co-located, their energy harvesting rates also vary significantly. Furthermore, although energy-harvesting sensor nodes usually are equipped with rechargeable batteries or super capacitors to

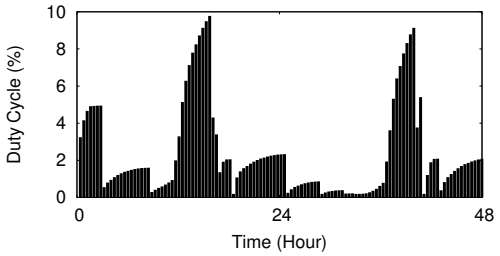


(a) Node 4

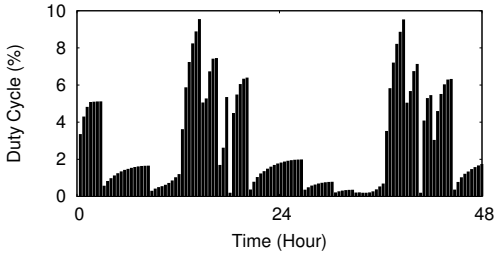


(b) Node 5

Figure 4.2: Harvested Power



(a) Node 4



(b) Node 5

Figure 4.3: Duty Cycle

alleviate the impact of energy variations, they are limited in energy storage capacity due to the small form factor requirement and waste energy unnecessarily due to the problem of energy leakage [74]. With such energy dynamics in the network, we can no longer schedule sensor nodes *a priori* as most existing battery-powered solutions do. Consequently, the question of how to effectively synchronize energy supply with sensor node activities while optimizing communication process, introduces a very demanding task.

Second, in the long run, ambient energy (e.g., solar, wind) is normally not intensive enough to sustain the continuous 100% duty cycle operation of sensor nodes [73, 74, 75]. Figure 4.3 shows the affordable duty cycles of node 4 and node 5, given the energy harvested in Figure 4.2. From Figure 4.3, we can see in order to ensure the continuous operation, the duty cycles of an energy-harvesting node can only range from 0.2% to 9.78%. From our empirical measurement results, even in a sunny day, the total energy harvested at a node can only supply itself to operate at 100% duty cycle for 6.37 hours. Essentially, during the operation of an energy-harvesting network, sensor nodes have to activate very briefly and stay in a dormant state for a very long period of time. In order to forward a packet in such always-dormant networks, a sender would experience

sleep latency - the time spent waiting for the receiver to wake up [76]. Moreover, since communication links between low-power sensor devices are usually unreliable [42], it brings further challenges in managing communication in sustainable sensor networks.

To the best of our knowledge, no prior work has studied the impact of *energy synchronization* on *low-duty-cycle networks with unreliable links*. We contribute this research direction with (i) theory, (ii) architecture and (iii) design.

4.3 System Models and Assumptions

Before presenting ESC in detail, we introduce the models and assumptions used in this work. In addition, we elaborate on the packet delivery process and define sleep latency in low-duty-cycle sensor networks. To simplify our description, we introduce our ESC design assuming (i) neighboring nodes are synchronized in the unit of a time instance and, (ii) there is at most one packet transmission within such a time instance. Later on in Section 4.7, we discuss how we can relax those assumptions in practice.

4.3.1 Working Schedule

The working schedule of a sensor node denotes the active-dormant behaviors of the sensor node over its lifetime. It consists of a set of *active instances*, during which a node can receive packets. Each active instance j at node i can be represented by a tuple (t_j^i, d_j^i), where t_j^i denotes the starting time of the active instance and d_j^i denotes the corresponding duration of the active instance j. Since many sensor node working schedules are periodic [77, 78, 79], it is sufficient to represent an infinite sequence of active instances, using repeated subsequences with a period time T. Let Γ_i be the working schedule of node i and the number of active instances within a period be N, we can have $\Gamma_i = \{(t_1^i, d_1^i), (t_2^i, d_2^i), ..., (t_N^i, d_N^i)\}$. According to its working schedule, a node continuously transits its state between active and dormant state. The duty cycle of node i, therefore is $\frac{\sum_{j=1}^{N} d_j^i}{T}$. For example, Figure 4.4 shows a periodic working schedule $\Gamma_i = \{(1,1), (5,2), (8,1)\}$ with a period time 10. The duty cycle of node i here is $\frac{4}{10} = 40\%$.

Figure 4.4: A Working Schedule

To simplify our description, in the rest of the chapter we assume all active instances have the same durations (τ). When a node is said to be active at time t, it has an active instance that starts at time t with duration of τ. We note that this definition

of working schedule can actually accommodate active instances with varying durations. Essentially, if we let τ be the finest granularity of time durations in the design, we can represent any node schedule with the fixed τ.

4.3.2 Network Model

We assume a network with N sensor nodes. At a given point of time t, a sensor node is in either an active or a dormant state. When a node is in the active state, it can receive packets transmitted from neighboring nodes. When a node is in the dormant state, it turns off all function modules except a timer (for the purpose of waking itself up). In other words, a node can *wake up to transmit a packet at any time, but can receive packets only when it is in its active state.* Since a node can only receive packets during its active state, the *packet ready time* at a node (i.e., the time a node receives a new packet and ready to forward to the next hop node), therefore *is the same as active instances in its working schedule.*

For a sensor node, its neighbors consistently transit between active and dormant states, and thus connectivity between a pair of nodes varies over time and becomes time-dependent. Formally, for a given time t, we denote the network topology as $G(t) = (V, E(t))$, where V is a set of N nodes within the network, and $E(t)$ is a set of directed edges at time t. An edge e(i,j) belongs to $E(t)$ if and only if (i) node i is within the communication range of node j, and (ii) node j is active and hence able to receive packet at time t. Essentially $G(t)$ represents the potential traffic flow within the network at time t.

4.3.3 Sleep Latency in Low-Duty-Cycle Network

In connected networks, a one-hop packet delivery latency usually includes processing delay, transmission delay, and propagation delay, which are normally in the order of milliseconds. In low-duty-cycle sensor networks, however, a sender may need to wait for its receiver to wake up before it can send a packet. We define *sleep latency* as the time duration from the moment a packet is ready at the sender to the moment the destined one-hop receiver is active. Sleep latency is usually in the order of seconds, which is much longer than other delivery latencies. Therefore, in this chapter we only consider

sleep latency for measuring E2E delay. In a network with perfect links, the E2E delay is the sum of sleep latencies along the path of data delivery.

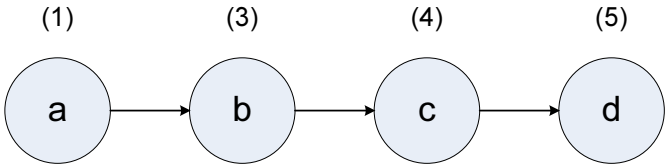


Figure 4.5: A Linear Network

To further illustrate the concept of sleep latency, we use the linear network shown in Figure 4.5 as an example. In Figure 4.5, the active instance of each node is labeled on top of each node. Assume node a has a packet ready to be sent at time 1, given that the node b wakes up at time 3, the sleep latency for the first attempted transmission from node a to node b therefore is $3 - 1 = 2$.

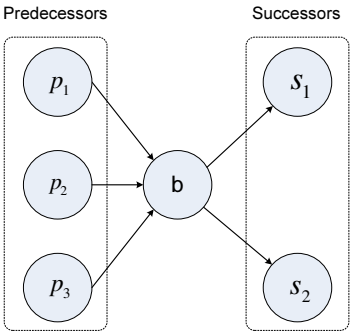


Figure 4.6: Predecessors and Successors

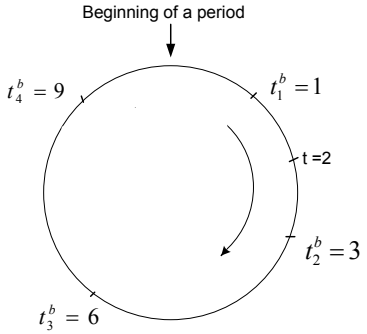


Figure 4.7: A Cyclic Working Schedule

4.4 Modeling of Cross-Traffic Delay

This section presents the model for cross-traffic delay at a node. In low duty-cycle networks, a packet is transmitted only when the receiver is in the active state, and nodes in such networks only activate very briefly and stay in the dormant state for the majority of time. Thus here we assume data traffic/congestion is low, which holds well for existing low-duty-cycle sensor networks [76, 37, 78].

4.4.1 Cross-Traffic Pattern

ESC is designed to be a flexible middleware between the network layer and data link layer, so that it can be used to support various existing routing protocols. Therefore, it is important to have a delay model that can capture the behavior of cross-traffic (many-to-many), a generalized case for one-to-one, many-to-one, and one-to-many traffic.

For a node b in the network, depending on the specific routing protocol adopted, there may be a set of nodes that forwards packets to node b, and we call those nodes *predecessors* of node b. Similarly, for different final destinations or multi-path routing protocols, node b would forward its packets to a certain set of nodes, and we call those nodes *successors* of node b. For example, in Figure 4.6, the predecessors of node b include node p_1, p_2 and p_3, while the successors of node b are nodes s_1 and s_2. Here we note that a node can be both a predecessor and a successor of node b, if this node exchanges data bidirectionally with node b.

To model the cross-traffic delay related to node b, we consider the expected delays for packets from all predecessor nodes through node b, then to corresponding successor nodes.

4.4.2 Delay Modeling

Assume at a predecessor node p_1, a packet destined to node b is ready at time t, where $t \in [0, T)$. Since the radio link between a pair of nodes is usually not perfect, node p_1 may need to initiate multiple transmissions before the packet has successfully arrived at node b. In order to obtain a corresponding sleep latency for each attempted transmission at node p_1 after packet ready time t, we introduce a cycle representation of a node working schedule shown in Figure 4.7.

In Figure 4.7, the cycle is equally divided by T ticks. Beginning at the 12 o'clock position, the time increases from 0 to T clockwise. Consequently, we can easily label the sequence of active instances at node b on the cycle. To measure the sleep latency (denoted as $L_t^b(k)$) introduced by node b, for a given k^{th} attempted transmission at time t, we can simply start from time t, follow the clockwise direction and measure the total distance traversed by visiting k labels after time t on the cycle. For example, as depicted in Figure 4.7 where T is 10 units of time, the packet is ready at time $t = 2$ and node b wakes up during time 1, 3, 6 and 9. For the first attempted transmission ($L_t^b(1)$), the corresponding sleep latency is 1 since the total time traversed for visiting one label (t_2^b) from time 2 is 1. Similarly, for the fourth attempted transmission ($L_t^b(4)$), the sleep latency is total time traversed from t to t_2^b, then to t_3^b, t_4^b and the fourth label t_1^b, which gives $1 + 3 + 3 + 2 = 9$ in total.

Link Reliability: Let the bi-directional link quality p_{ab} denote the success ratio of a round-trip transmission (DATA and ACK) between node a and node b. The probability that packet transmission succeeds at the k^{th} attempt is the probability that previous $k - 1$ attempts failed times the probability that the current attempt succeeds, which is simply the link quality p_{ab}. Therefore the probability that the packet reaches node b from node a at its k^{th} attempt can be expressed as $(1 - p_{ab})^{k-1} p_{ab}$.

Assuming the maximum number of packet retransmissions within the network is R_{max}, for the packets arriving at node b from a, the probability that they arrive at k^{th} attempt is under the condition that the packet is delivered within R_{max} retransmissions.

The probability that a packet is delivered within R_{max} retransmissions can be expressed as $1-(1-p_{ab})^{R_{max}}$, and the corresponding conditional probability can be written as:

$$P^{ab}(k) = \frac{(1-p_{ab})^{k-1}p_{ab}}{1-(1-p_{ab})^{R_{max}}} \tag{4.1}$$

Delay Over a Single link: For a packet ready time t at node a, the expected transmission delay to reach node b is the sum of the product of probability that the packet reaches node b at its k^{th} attempt and corresponding sleep latency. Consequently, it can be formulated as:

$$D_{ab}(t) = \sum_{k=1}^{R_{max}} P^{ab}(k)L_t^b(k) \tag{4.2}$$

Delay from One Predecessor to One Successors: For a packet ready time t at a predecessor node p_i, assuming the packet arrives at node b at the k^{th} attempted transmission, its corresponding delay is simply $L_t^b(k)$. Upon receiving the packet at time $t+L_t^b(k)$, node b would forward the received packet to a destined successor node s_j with a delay of $D_{bs_j}(t+L_t^b(k))$. Then the expected delay from predecessor p_i to successor s_j with packet ready time t at predecessor p_i is the product of probability that the packet arrives node b at its k^{th} attempted transmission and corresponding cross-traffic delay, which can be expressed as:

$$D_{p_is_j}(t) = \sum_{k=1}^{R_{max}} P^{p_ib}(k)(L_t^b(k) + D_{bs_j}(t+L_t^b(k))) \tag{4.3}$$

Delay from Multiple Predecessors to Multiple Successors: To model the expected cross-traffic delay at node b from all packet ready times at all predecessor nodes to all successor nodes, node b needs to know the portion of traffic that reaches node b from each packet ready time at all predecessor nodes to all successor nodes. To obtain those statistics, each predecessor of node b can piggyback packet ready time for each sent packet. Then at node b, with known sender and packet ready time for each packet, it can easily keep track of what percentage of packets is from a given packet ready time t at a predecessor node p_i, to a specific successor node s_j. Assume the number of packet ready time at a predecessor node p_i is N_{p_i}, for each packet ready time $t_k^{p_i}$, we can represent the percentage of traffic from a packet ready time $t_k^{p_i}$ at a predecessor node p_i to

a successor node s_j as $W_k^{p_i s_j}$. Let the number of predecessor nodes and successor nodes at node b be N^p and N^s, respectively, we can express the expected delay of cross-traffic at node b as:

$$D_b = \sum_{i=1}^{N^p} \sum_{j=1}^{N^s} \sum_{k=1}^{N_{p_i}} W_k^{p_i s_j} D_{p_i s_j}(t_k^{p_i}) \tag{4.4}$$

4.5 Energy Synchronization Control

With the cross-traffic delay model available, we now introduce energy synchronization control for minimizing communication delay in sustainable sensor networks. In this section, we first assume the working schedules of predecessors and successors of a node are known and up-to-date. Later in Section 4.6, we discuss the impact of obsolete schedules and methods to keep the schedules up-to-date.

4.5.1 Main Idea

As shown in Section 4.2, energy harvested from surrounding environments varies significantly over time at a sensor node. In order to make full use of the available energy supply, we need to enhance system performance when there is abundant scavenged energy available; conversely, we need to decrease duty-cycles with a minimum performance degradation when there is a shortage in the power supply. Previous works [80, 81] have demonstrated methods for deciding appropriate duty-cycle of a sensor node with in-situ energy supply. In this section, we further focus on the impact of duty-cycle changes on communication delay. More specifically, when we increase the duty-cycle of a node with additional available energy, we aim at minimizing cross-traffic delay at the node. Similarly, when a node needs to decrease its duty-cycle due to lack of sufficient energy supply, we would like to achieve a minimum increase in the cross-traffic delay of the node. As a result, harvested energy at an individual node is synchronized to minimize the communication delay within the network.

4.5.2 Decrementing Single Active Instance

For a given node b, assume its current working schedule is $\Gamma_b = \{t_1^b, t_2^b, ..., t_{N_b}^b\}$ and current energy supply can only afford $N_b - 1$ active instances to guarantee the sustainability of the node. Therefore, we need to remove one active instance from node b's current working schedule such that the increase of cross-traffic delay at node b is minimized. Since sustainable sensor nodes work in extremely low duty-cycles, the number of active instances in its working schedule is very small and would always below a constant value c. Consequently, to find the optimal active instance for decrement, we can simply test

all existing active instances and select the one that yields the minimum delay at node b. The complexity of the optimal single active instance decrement, therefore, is just $\mathcal{O}(1)$.

4.5.3 Augmenting Single Active Instance

Intuitively, for augmenting one active instance at a node, we can perform an exhaustive search on all time instances within a period time T and choose the time instance that yields a minimum cross-traffic delay at a node. Although this naive algorithm is acceptable, a much more efficient algorithm can be designed based on the stair effect as presented in this section.

For a given node b, we can divide its period time into multiple intervals according to active instances of its predecessors and successors. For example, as shown in Figure 4.8, assume the predecessor and successor of node b is node p and node s, respectively. Let the active instances at node p and node s be $\{t_1^p, t_2^p\}$ and $\{t_1^s, t_2^s\}$, respectively. According to their locations on the cyclic working schedule, we can easily obtain four intervals, namely: (t_1^p, t_1^s), (t_1^s, t_2^p), (t_2^p, t_2^s) and (t_2^s, t_1^p).

Intuitively, one would expect the timing of an augmented active instance within an interval to yield different cross-traffic delays at node b. However this is not the case. We observe that *given schedules of predecessors and successors of node b, cross-traffic delay at node b only depends on the* **counts**, *instead of* **timing**, *of active instances within each interval.*

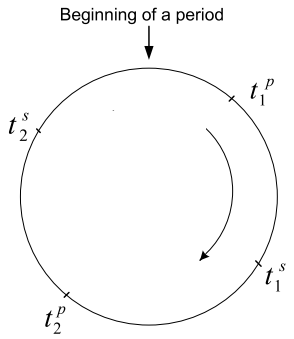


Figure 4.8: Period Partition Example

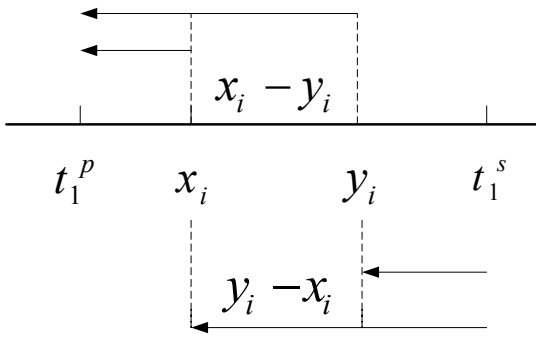


Figure 4.9: Delay Difference Example

To validate this observation, it is sufficient to prove that for two arbitrary layouts of active instances of same size within an interval, the cross-traffic delays at node b are

the same if packets are received within this interval. Following is the formal proof.

Lemma 4.5.1 *let $X = x_1, x_2, ..., x_n$ and $Y = y_1, y_2, ..., y_n$ be two sets of active instances of node b within one interval (where $x_1 < x_2 < ... < x_n$ and $y_1 < y_2 < ... < y_n$), and let D^X and D^Y be corresponding cross-traffic delays.* ***If*** *$|X| = |Y|$,* *$D^X = D^Y$.*

Proof For any packet ready time t at a predecessor node p, since $x_1, x_2, ..., x_n$ and $y_1, y_2, ..., y_n$ are in the same interval, as shown in Figure 4.9, the difference of sleep latencies for the i^{th} attempted transmission for X and Y after time t is just the difference of their respective active instances. Consequently, $L_t^X(i) - L_t^Y(i) = x_i - y_i$, for any $1 \leq i \leq n$.

Similarly, since x_i and y_i are in the same interval, the j^{th} attempted transmission from either x_i or y_i reaches the same active instance at a successor node s. Clearly shown in Figure 4.9, $L_{x_i}^s(j) - L_{y_i}^s(j) = y_i - x_i$, where $1 \leq i \leq n$ and $1 \leq j \leq R_{max}$.

By applying Equation 4.3 to X and Y, we have,

$$\begin{aligned} & D_{ps}^X(t) - D_{ps}^Y(t) \\ = & \textstyle\sum_{i=1}^{R_{max}} P^{pb}(i) \\ & (L_t^X(i) - L_t^Y(i) + D_{bs}(L_t^X(i)) - D_{bs}(L_t^Y(i))) \\ = & \textstyle\sum_{i=1}^{R_{max}} P^{pb}(i)(x_i - y_i + D_{bs}(x_i) - D_{bs}(y_i)) \end{aligned}$$

Since,

$$\begin{aligned} D_{bs}(x_i) - D_{bs}(y_i) &= \sum_{j=1}^{R_{max}} P^{bs}(j)(L_{x_i}^s(j) - L_{y_i}^s(j)) \\ &= \sum_{j=1}^{R_{max}} P^{bs}(j)(y_i - x_i) \end{aligned}$$

As $\sum_{j=1}^{R_{max}} P^{bs}(j) = 1$ we have:

$D_{bs}(x_i) - D_{bs}(y_i) = y_i - x_i$

Consequently,

$$D^X_{ps}(t) - D^Y_{ps}(t) = \sum_{i=1}^{R_{max}} P^{pb}(i)(x_i - y_i + y_i - x_i) = 0$$

As a linear combination of $D^X_{ps}(t) - D^Y_{ps}(t)$ for all packet ready times at all predecessor nodes, we have $D^X - D^Y = 0$.

According to Lemma 4.5.1, for augmenting one active instance within a partitioned time interval, the cross-delay at node b is not affected by the timing of the augmented active instance. In other words, single active instance augmentation yields the same cross-traffic delay at node b within each of partitioned time intervals. Consequently, for a single active instance augmentation, we have a *stair effect of cross-traffic delays at node b* within each partitioned time interval. Based on this counterintuitive observation, we can have the following theorem on finding the optimal single active instance augmentation in the constant time.

Theorem 4.5.2 *Assuming there are c intervals partitioned by active instances at predecessors and successors of a node and x_i is a random active instance within interval i, where $i = 1, 2, ..., c$. Let D^j represent the cross-traffic delay at the node after augmenting an active instance j to its original working schedule, and $Min(D^{x_i}) = D^{x_k}$, where $i = 1, 2, ..., c$, then any time instance within interval k is the optimal single active instance augmentation and the complexity of this process is $\mathcal{O}(1)$.*

Proof Within any time interval i that is partitioned by active instances at predecessors and successors, for any random augmented active instance x_i, essentially we increase the number of active instances within the interval by one. By Lemma 4.5.1, it is straightforward that those augmented active instances yield the same cross-traffic delays at node b. Therefore, to find the optimal augmented active instance at a node, we just need to check the cross-traffic delay with a random active instance augmentation within each of time intervals, and find the interval k that yields the minimum cross-traffic delay.

Since sustainable sensor networks operate in extremely low duty-cycle, we would have a very few packet ready times from predecessor nodes and active instances from successor nodes. Consequently, there would be only a few time intervals that need to be checked in order to find the optimal augmented active instance. In other words, the number of time intervals c is a constant value, and thus the complexity of finding optimal active instance is just $\mathcal{O}(1)$.

Algorithm 3 Active Instance Augmentation at a Node b

Input: Working schedule Γ for predecessor and successor nodes
Input: Real-time link quality for cross-traffic from predecessor and successor nodes
Input: Traffic distribution for cross-traffic
Output: The optimal augmented active instance

```
1: Sort active instances for predecessor and successor nodes and form c time intervals
2: D_min ← ∞
3: X ← ∅
4: for i ← 1 to c do
5:    x_i ← random active instance within time interval i
6:    Augment active instance x_i to working schedule Γ_b of node b
7:    Calculate cross-traffic delay D_b using Equation 4.4
8:    if D_b < D_min then
9:       D_b ← D_min
10:      X ← x_i
11:   end if
12:   Remove active instance x_i from working schedule Γ_b of node b
13: end for
14: return X
```

Guided by Theorem 4.5.2, we can derive the detailed energy synchronization process at a node b shown in Algorithm 4.5.3. First, based on working schedules of predecessor and successor nodes of node b, we create c time intervals and initialize system variables (Line 1 to Line3). Then for each time interval, we select a random time instance x_i, augment it to working schedule of node b and calculate cross-traffic delay D_b (Line 5 to Line 7). For each calculated D_b, we compare it to current minimal cross-traffic delay D_{min} and store new minimal delay value and the corresponding time interval if necessary (Line 8 to Line 11). After testing each time interval, we obtain the optimal active instance augmentation for node b. Since time complexity of our energy synchronization process is just $\mathcal{O}(1)$, it incurs little energy overhead for a node b to perform such procedure in real-time.

To further illustrate Theorem 4.5.2 and Algorithm 4.5.3, we give an example in Figure 4.10. Assuming one period time contains 200 time instances, Figure 4.10 shows the expected cross-traffic delay at a node for different augmented active instances. For example, the delay corresponding to active instance 1 represents the delay at node b after augmenting an active instance at time 1. The node shown in Figure 4.10 has

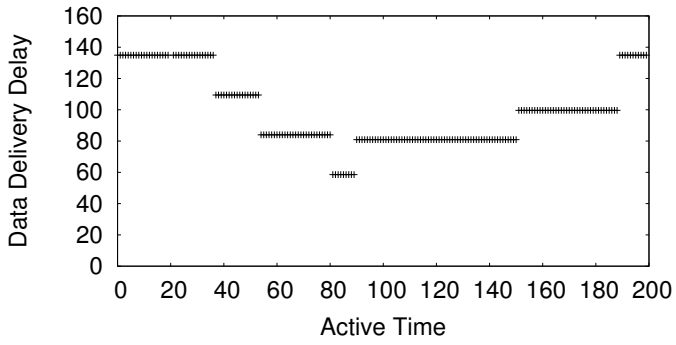


Figure 4.10: Stair Effect of Cross-Traffic Delay

a predecessor with active instances $(36, 53, 80)$ and a successor with active instances $(90, 151, 189)$. According to our analysis above, we can divide one period time into the following intervals: $(36, 53)$, $(53, 80)$, $(80, 90)$, $(90, 151)$, $(151, 189)$, $(189, 36)$.

From Figure 4.10, it is clear that we have a stair effect of delays at the node among the above time intervals, which is consistent with our analysis. The optimal augmented active instance, therefore, is any value within interval $(80, 90)$.

4.5.4 Bursty Augmentation and Decrement

In the previous two subsections, we introduced the optimal solutions for augmenting and decreasing *a single active instance* for the scenario that energy variation is slow. However, the change in harvested energy could be bursty and therefore a node may need to increase or decrease multiple active instances simultaneously. In this section, we present a minimal cost solution for augmenting and decreasing multiple active instances.

A straightforward exhaustive search for multiple active instances augmentation and decrement is no longer acceptable since the computational complexity grows exponentially with the number of augmenting or decreasing active instances. Therefore, a low-cost solution that guarantees optimality of multiple active instances augmentation and decrement is desirable. Fortunately, we observe that multiple active instances augmentation and decrement can be *optimally* solved using a greedy selection.

The main idea of greedy solution is that by applying single active instance augmentation or decrement n times, we can obtain the optimal solution for augmenting or decreasing n active instances at a node. The detailed proof is shown in Appendix B.

Since the computational complexity of single active instance augmentation or decrement is $\mathcal{O}(1)$, the complexity for augmenting or decrementing n active instances is $\mathcal{O}(n)$.

4.6 Maintaining Logical Connectivity

In the previous section, energy synchronization control is designed to adjust the working schedules at receiving nodes, assuming the schedule of predecessors and successors are known and up-to-date. In this section, we investigate the impact of obsolete schedules and how to maintain connectivity while updating schedules.

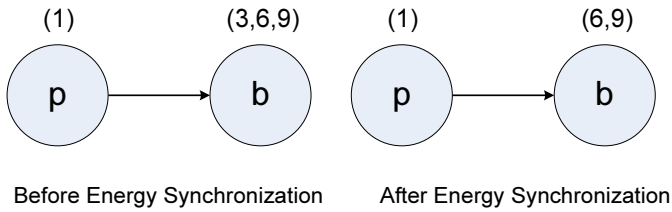


Figure 4.11: Impact of Energy Synchronization Example

4.6.1 Impact of Schedule Updates

To understand how obsolete schedules can affect the connectivity between nodes, let us start with the example shown in Figure 4.11, where the working schedule of node b has been changed but has not yet been updated to node p:

- **Unnecessary loss:** If node b decreases its duty-cycle due to insufficient energy supply, then one or more original active instances at node b may be removed. However, before node p is updated, it would continue to deliver packets according to the old (obsolete) working schedule of node b, which could suffer significant greater packet loss than necessary. For example, as shown in Figure 4.11, node b reduces its duty-cycle by changing its working schedule from $(3, 6, 9)$ to $(6, 9)$. Assuming predecessor node p has a packet ready at time 1, unaware of the new working schedule at node b, node p would try to deliver the packet with active instance sequence $(3, 6, 9, 3, 6, 9, ...)$ until the packet is successfully delivered or the number of retransmissions reaches the bound R_{max}. Since node p would always fail at attempted transmissions at removed active instance 3, the data delivery ratio therefore is clearly decreased.

- **Suboptimal delay:** Similarly, if node b increases its duty-cycle and a predecessor node is unaware of the augmented active instances at node b, it would continue to

deliver the packets at node b's original active instances rather than take advantage of the augmented active instances at node b to reduce the delivery delay. In addition, for both predecessors and successors, the new schedule of node b is essential for them to perform effective energy synchronization. Therefore, it is crucial for node b to promptly disseminate its new working schedule to its predecessor and successor nodes.

4.6.2 Shuffle-Based vs. Adjustment-Based Energy Synchronization

There are two general approaches that can be taken when node b increases or decreases its duty cycles. Node b can either generate a complete new working schedule with a given new energy budget (termed a *shuffle*), or node b can increase or decrease its duty-cycle on top of its previous working schedule (termed an *adjustment*). For example, assume node b currently has active instances $(3, 6)$ and the current energy supply could afford to increase one more active instance. A shuffle would generate a completely new schedule solely based on the packet ready time at predecessors and active instances at successor nodes. Therefore, the new working schedule at node b after one active instance augmentation could be $(2, 7, 9)$, which has no overlap with the previous schedule. On the other hand, an adjustment would add the augmented active instance to the previous schedule. Consequently, the new working schedule produced by the adjustment could be $(3, 6, 8)$, where $(3, 6)$ are identical to the previous working schedule.

One interesting phenomena of duty-cycled sensor networks is the separation between *physical connectivity* and *logical connectivity*. Two nodes are *physically connected* if they are within each other's communication range and *logically connected* only if they can communicate. Unlike traditional networks, a low-duty-cycle network could be physically connected, but logically partitioned if nodes do not know each other's working schedules.

Let logical connectivity ℓ_{ab} be the packet delivery ratio between two nodes a and b, after R_{max} retransmissions. Let Γ_b be the set of first R_{max} active instances in the original schedule and Γ'_b be the new schedule of node b. The logical connectivity ℓ_{ab}, therefore, is:

$$\ell_{ab} = 1 - (1 - p_{ab})^K \; where \; K = |\Gamma_b \cap \Gamma'_b| \tag{4.5}$$

Shuffle

In the case of a complete shuffle, $|\Gamma_b \cap \Gamma'_b| = 0$, therefore, logical connectivity $\ell_{ab} = 0$. In other words, node a and b are disconnected. In the case of a partial shuffle, node a experiences the long packet delivery delay as well as low packet delivery ratio before it receives the new schedule from node b. This is because without knowing the new working schedule of node b, node a tries to deliver the data during node b's original active instances. However, most of the original active instances no longer exist due to the shuffling. Consequently, a packet might experience a long delivery delay (after several pointless transmissions), or even get dropped because it exceeded the maximum number of retransmissions.

On the other hand, a shuffle creates an optimal working schedule at node b with the known packet ready times at predecessors and active instances at successors. After the new schedule is updated, a shorter delay is expected.

Adjustment

In the case of an adjustment, $|\Gamma_b \cap \Gamma'_b| = \min\{|\Gamma_b|, |\Gamma'_b|\}$, the logical connectivity is optimally maintained. Adjustment preserves the previous schedule when the duty-cycle increases, while it keeps the majority of the previous schedule when the duty-cycle decreases. Consequently, before a predecessor receives the new schedule of node b for the increased duty-cycle, its packet delivery delay and delivery ratio would remain unchanged. Similarly, for the decreased duty-cycle, the packet delivery delay for a predecessor unaware of the new schedule of node b would be consistent with the scenario when the predecessor knows the new schedule since in either case the predecessor would not be able to deliver the data during the removed active instances at node b. The packet delivery ratio, however, would be slightly reduced since the predecessor wastes a certain number of attempted transmissions at the time instance that node b is no longer awake to receive the packet.

Comparison

Since adjustment produces smaller delays before the new working schedule reaches predecessor nodes and shuffle creates smaller delays after the new working schedule reaches

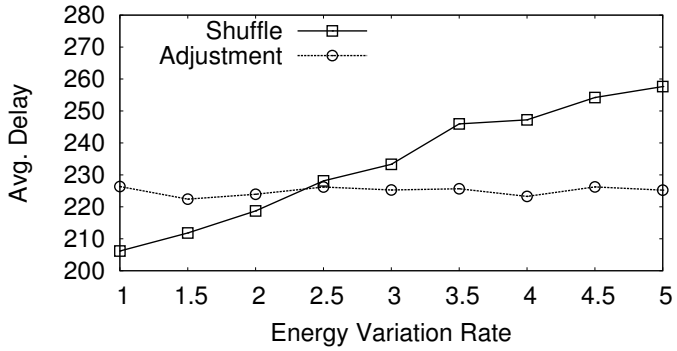


Figure 4.12: Delay vs. Energy Variation Rate

predecessor nodes, the overall delay at a node therefore is influenced by the energy variation rate at the node. Figure 4.12 shows average cross-traffic delays for both adjustment and shuffle under different energy variation rates. In Figure 4.12, the energy variation rate denotes the average number of energy changes over a period of 100,000 units of time. When the energy variation rate is low, the delay after schedule dissemination dominates the overall delay and shuffle therefore has a smaller overall delay than that of adjustment. For example, when the energy variation rate is 1, the overall delay for adjustment and shuffle is 226.34 and 206.17 units of time, respectively. As the energy variation rate becomes larger, the delay before schedule dissemination weights more in overall cross-traffic delay. Consequently, adjustment that has a smaller delay before schedule dissemination also has a smaller overall delay than shuffle. As shown in Figure 4.12, from energy variation rate 2.5 to 5, adjustment has smaller delay than that of shuffle. At energy variation rate 5, the delay for adjustment and shuffle is 225.22 and 257.65, respectively. This study indicates that the design options on ESC should be decided based on how fast ambient energy changes over time.

4.7 Practical Issues

This section completes the description of our ESC design by discussing several practical design issues, such as time synchronization and multiple transmissions within a time instance.

4.7.1 Low-cost Time Synchronization

For the sake of clarity, we introduce ESC design in a synchronized mode. Clearly, the operation of ESC depends on neither neighbor time instance nor global synchronization. It is sufficient for ESC only knows the *wake-up time interval* of predecessor and successor nodes. To know those wake-up time intervals, simple and low-cost local synchronization techniques [29] can achieve an accuracy of $2.24\mu s$ with the cost of exchange a few bytes of packets among neighboring nodes every 15 minutes. Since an active instance is typically ranges from $2000\mu s$ to $20{,}000\mu s$, the accuracy of $2.24\mu s$ is far more than sufficient. In addition, ESC does not require the transmission starts at the beginning of an active instance, which further relaxes the requirement of accuracy for time synchronization.

4.7.2 Multiple Transmissions within a Time Instance

While describing our network model in Section 4.3, we assume there is at most one packet transmission during an active instance. This is true if nodes are equipped with slow radio. However, if fast radio are used, it is possible to transmit multiple packets within an active instance. To accommodate such scenario in modeling of cross-traffic delay, we can simply rewrite the bidirectional link quality between two nodes as $p'_{ab} = 1 - (1 - p_{ab})^m$, where m is the maximum number of transmissions allowed in an active instance. Essentially, the new p'_{ab} represents the probability that the receiver received the packet by m transmissions.

4.8 Implementation and Evaluation

In order to validate the performance and feasibility of ESC in practice, we have fully implemented ESC on the TinyOS/Mote platform in nesC. To compare the performance of ESC, we also implemented a random working schedule synchronization scheme that randomly adjusts active instances of the original node working schedule with increasing or decreasing node duty cycles.

4.8.1 Experiment Setup

During the experiment, we randomly placed 30 MicaZ nodes on our test-bed. The transmission power at MicaZ motes is tuned down to form a multi-hop network. Neighboring nodes are synchronized using FTSP [29] and each active instance lasts for $20ms$. The available energy budget over time at each node is derived from the actual energy harvesting rate measured at our running prototype and the corresponding energy harvesting model [74]. The range of duty cycle varies from around 0.2% to 10%. According to the available energy budget, each node turns on and off its radio based on the energy synchronized working schedule. To study the performance of ESC under various routing protocols, we also implemented link-quality-based ETX [35] and sleep-latency-based DESS [37] on motes.

4.8.2 Performance Comparison

In this section, we compare the E2E delay for both ESC and the randomized scheme. During the experiment, for each routing protocol, over 1000 packets are transmitted from a random source to a random destination. To ensure the fair comparison, ESC packets and randomized energy synchronization packets are sent alternatively to minimize the impact of energy variation and link quality fluctuation.

In Figure 4.13 we study the E2E delay for ESC and the randomized scheme under ETX. Clearly, ESC performs much better than the randomized scheme. While 80% of ESC packets reach their destinations within 877ms, the corresponding percentile for the randomized scheme is 1451ms, which is about 65% increase. Since ETX picks the route with the minimum number of expected transmissions, the performance gap between

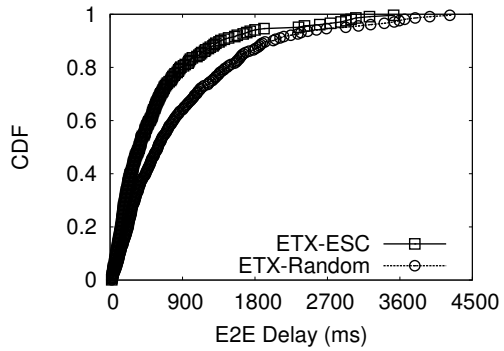


Figure 4.13: ETX E2E Delay

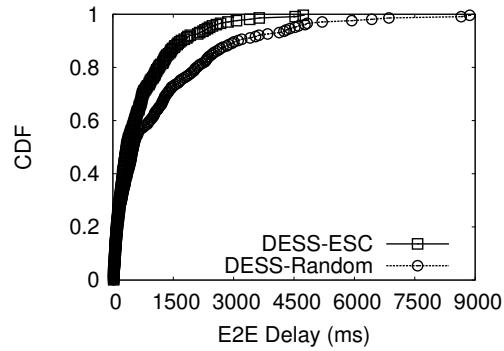


Figure 4.14: DESS E2E Delay

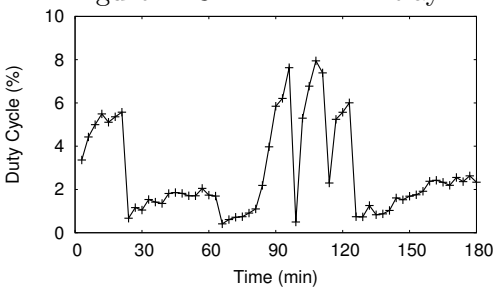


Figure 4.15: Node Duty-Cycle Over Time

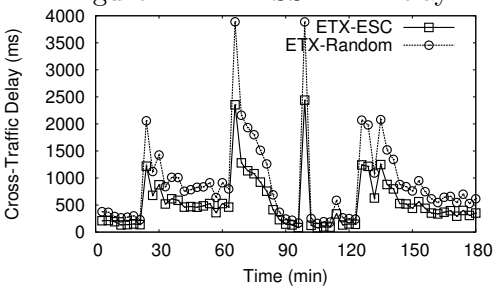


Figure 4.16: Cross-Traffic Delay Over Time

ESC and the randomized scheme therefore is mainly due to the minimized cross-traffic delay for ESC.

Similar to the results for ETX, ESC also significantly outperforms the randomized scheme under DESS. While the 80% percentile for ESC is 1131ms, the corresponding number for the randomized scheme is 2091ms, which almost doubles the number of ESC. In addition, the randomized scheme has much longer tail than ESC. While all messages for ESC reach the destinations within 6299ms, the longest E2E delay for the randomized scheme is 12343ms. The reason for such long tail of the randomized scheme is because the penalty of a failed transmission for the randomized scheme is much larger than the ESC, as ESC has carefully scheduled the radio activity to minimize the impact of the failed transmissions.

To further reveal the performance of ESC over time dimension, Figure 4.15 and Figure 4.16 show the duty-cycle of a deployed node and its corresponding cross-traffic delay under ESC and the randomized scheme over a period of 3 hours. By comparing Figure 4.15 and Figure 4.16, we can see the cross-traffic delay matches the available

duty-cycle well. For example, the peaks of delay occur when the node duty-cycle drops to around 0.4% at time 65 and 100. In addition, although both ESC and the randomized scheme react to the duty-cycle change promptly, the cross-traffic delay for ESC is always smaller than that of the randomized scheme. This consistent smaller cross-traffic delay of ESC over time further explains the smaller E2E delay for ESC in Figure 4.13.

4.9 Simulation Evaluation

In addition to test-bed evaluation in Section 4.8, to understand the system performance of ESC under numerous network settings, in this section we provide simulation results with over 1000 sensor nodes. To investigate the flexibility of ESC design, we choose two state-of-the-art solutions as underlying routing protocols:

- Link-Quality-based: ETX [35] in MobiCom'03
- Sleep-Latency-based: DESS [37] in INFOCOM'05

4.9.1 Simulation Setup

In the simulation, except where otherwise specified, we deploy up to 1,200 sensor nodes randomly in a 400m×400m square field. A sink is positioned in the center of the deployment field, and each sensor node sends its packet to the sink over multiple hops. The radio model was implemented according to [53], which considers the oscillation nature of the radio links and has several adjustable parameters. During the simulation, we set these parameters strictly according to the CC2420 radio hardware specification [57].

Each experiment was repeated 100 times with different random seeds, node deployments, and node working schedules. Data collected at each node were obtained by averaging over 10000 source-to-sink communications. The 95% confidence intervals are within 1∼4% of the means.

4.9.2 System Performance Over Time

In this section, we reveal the effectiveness of ESC over time in terms of communication delays. Figure 4.17 shows the average cross-traffic delay at a node over a period of 25,000 units of time. At time 0, active instances are allocated randomly within nodes (hence not optimally). The node increases its duty-cycle at time 5,000 and 10,000 and decreases its duty-cycle at time 15,000 and 20,000. It is clear that after the node increases its duty-cycle at time 5,000, the delay at the node significantly drops. For example, within time interval $[0, 5000]$, the average delay is 249.26 units of time. In contrast, during time $[5000, 10000]$, the average delay drops to 90.51, which is around only 36.3% of the original delay. After increasing the duty-cycle again at time 10,000, the delay at

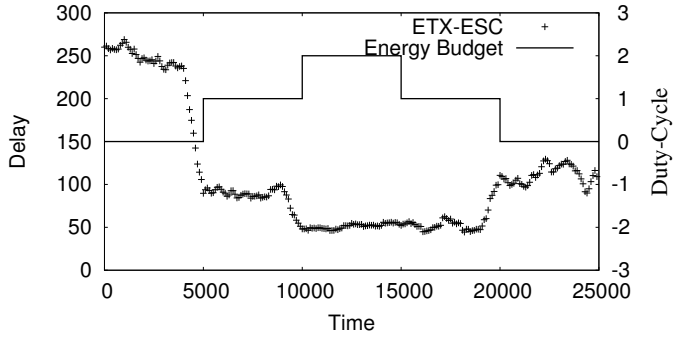


Figure 4.17: Delay Over Time

the node further reduces to 50.32 during time $[10000, 15000]$, almost half the previous delay. When the duty-cycle decreases at time 15,000, the average delay only slightly increases to 51.36 units of time within time interval $[15000, 20000]$. Finally, when we further reduce the duty-cycle at time 20,000, the delay increases to 113.66 which is only around 45.6% of the initial delay during time $[0, 5000]$, when allocation is not optimal. From this figure, it is clear that ESC effectively reduces the delay at the node when its duty-cycle increases while it minimally increases the delay when the node decreases its duty-cycle, converging gradually from an initial not-optimal allocation into an optimal allocation.

4.9.3 Impact of Node Densities

In this section, we examine the impact of node densities on E2E delay for both ETX and DESS networks with varying energy supplies over time.

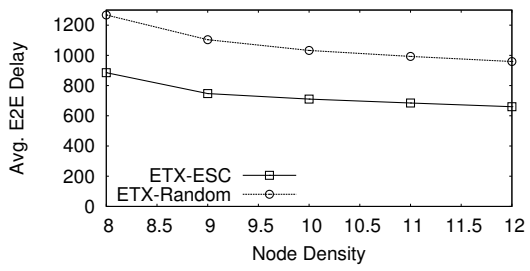


Figure 4.18: ETX Delay vs. Node Density

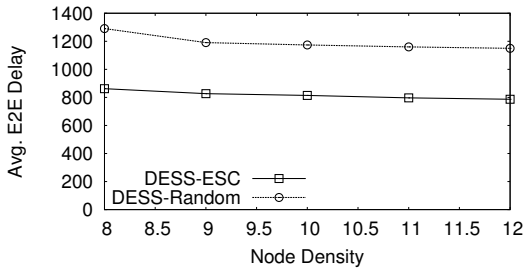


Figure 4.19: DESS Delay vs. Node Density

During the simulation, for each node both ESC and the randomized energy synchronization scheme described in Section 4.8 have the same energy supply over time to ensure fair comparisons. As can clearly be seen from both Figure 4.18 and Figure 4.19, ESC has a much smaller delay than the Randomized scheme at all node densities for both ETX and DESS. For example, at node density 10, ETX-ESC has a delay of 710.64 units of time while ETX-Random has a delay of 1032.51 units of time, which is about 45% larger than the delay for ETX-ESC. Similarly for DESS at node density 10, the delay for DESS-ESC and DESS-Random is 814.02 units of time and 1173.95 units of time, respectively.

4.9.4 Impact of Node Duty Cycles

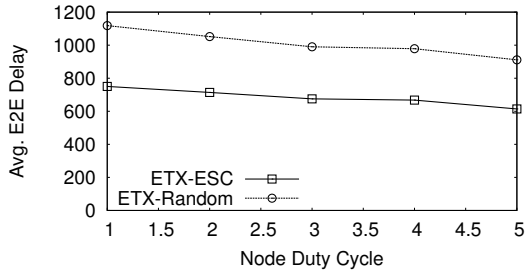


Figure 4.20: ETX Delay vs. Duty Cycle

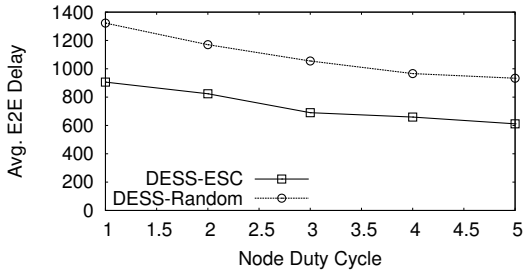


Figure 4.21: DESS Delay vs. Duty Cycle

In this section, we study the impact of node duty cycles on E2E delay for applying ESC to ETX and DESS in energy-varying sustainable sensor networks. In both Figure 4.20 and Figure 4.21, we can see that ESC outperforms randomized scheme at all node duty cycles. As the node duty cycle increases, E2E delays for both ETX and DESS under ESC and randomized scheme are decreasing. This is because with a higher node duty-cycle in the network, the sleep latency between a sender and a receiver is reduced, as there are more active instances at the receiver to receive incoming packets from sending nodes. For example, average E2E delays for ESC-ETX decreases from 750.27 units of time to 614.15 units time while the delays for Random-ETX decreases from 1118.48 to 911.31 units of time.

4.10 Related Work

Several technologies have been developed to extract energy from the environment, including solar, motion, biochemical and vibrational [69, 71]. Building on those energy-harvesting technologies, researchers have designed various types of platforms to collect ambient energy from the environment with optimal efficiency [82, 70, 75]. To fully utilize the harvested energy and ensure the sustainable operation of sensor node, Kansal et al. [80, 73] and Vigorito et al. [81] have presented both theoretical and experimental results on deciding the appropriate working duty-cycle of sensor nodes with information on current energy harvesting rates. To further study the impact of energy leakage for energy-harvesting sensor networks, TwinStar system suggests node duty-cycle base on user specified lifetime and energy information including energy harvesting rate, remaining energy in the system and energy leakage rate [74].

On the other hand, with the growing gap between energy supply and demand recently, there has been a surge of interest on intermittently connected networks. For scenarios with mobile nodes, a number of effective solutions has been proposed to data communication in such networks [83, 84, 85]. For scenarios with low-duty-cycle nodes, by assuming perfect link qualities, both work [37, 48] introduce serval techniques for minimizing communication latency while providing energy-efficient periodic node working schedules. To address both low-duty-cycle and unreliable communication links, Gu et al. [76] introduce a dynamic switch-based forwarding using optimized forwarding sequences. Su et al. [78] propose both on-demand and proactive algorithms for routing packets in low-duty-cycle networks. More recently, serval efficient flooding protocols have been introduced to tackle the unique challenge in low-duty-cycle sensor networks [86, 87].

However, none of those prior works investigate how changing the duty-cycle of sensor nodes affects communication performance in sustainable sensor networks and how we can adaptively synchronize node working schedules with specified duty-cycle budgets. In this work, we advance state-of-the-art solutions for both energy-harvesting and low-duty-cycle sensor networks and providing effective methods of synchronizing node working schedules with varying duty-cycle budgets.

4.11 Conclusion

In this work, we reveal that cross-traffic delay through a duty-cycled node is determined only by the number of active instances at intervals, partitioned by active instances of predecessor and successor nodes. This allows us to design energy-synchronized control with $\mathcal{O}(1)$ time complexity for sustainable networks in which energy supplies and demands are in balance. In a low-duty-cycle network, updating neighbors' working schedules would be slow, leading to inconsistent views on active instances. To address this issue, we investigate the impact of obsolete working schedules on logical link quality and demonstrate the tradeoff between shuffle-based and adjustment-based allocation under different energy variation rates. Our evaluation demonstrates that ESC can effectively reduce delay and increase delivery ratios, while synchronizing radio activity with available energy.

Chapter 5

Conclusion and Discussion

In this dissertation research, we introduce a comprehensive design framework for extremely low-duty-cycle sensor networks including a generic sensing architecture, a novel, optimal data forwarding scheme under intermittent connectivity in the network, as well as duty-cycle adaptation under varying energy supply over time for energy-harvesting sensor networks.

Specifically, in Chapter 2, we propose a unified sensing architecture called uSense. It features an asymmetric design and supports various sensing coverage algorithms, along with a simple generic switching algorithm at each individual sensor nodes. In addition, we introduce a two-level global scheduling algorithm called uScan, which is seamlessly supported by the uSense architecture. In the first level, we propose an optimal scheduling algorithm in terms of minimizing area breach. In the second level, we propose a linear algorithm to address the set-cover problem when the layout of tiles satisfies certain conditions. With three novel ideas: *Asymmetric Architecture*, *Generic Switching* and *Global Scheduling*, our work has successfully achieved flexibility and efficiency for the sensor network coverage problem.

In Chapter 3, we introduce a dynamic switch-based forwarding (DSF) scheme for extremely low duty-cycle sensor networks, which addresses the combined effect of unreliable radio links and sleep latency in data forwarding. We derive a distributed model for data delivery ratio (EDR), E2E delay (EED), and energy consumption (EEC) at individual nodes and optimize the forwarding action in terms of these three metrics.

Both testbed experiments and large scale simulation results demonstrate that DSF significantly improves source-to-sink communication over several state-of-the-art solutions in low duty-cycle sensor networks with unreliable radio links.

In Chapter 4, we reveal that cross-traffic delay through a duty-cycled node is determined only by the number of active instances at intervals, partitioned by active instances of predecessor and successor nodes. This allows us to design energy-synchronized control with $\mathcal{O}(1)$ time complexity for sustainable networks in which energy supplies and demands are in balance. In a low-duty-cycle network, updating neighbors' working schedules would be slow, leading to inconsistent views on active instances. To address this issue, we investigate the impact of obsolete working schedules on logical link quality and demonstrate the tradeoff between shuffle-based and adjustment-based allocation under different energy variation rates. Our evaluation demonstrates that ESC can effectively reduce delay and increase delivery ratios, while synchronizing radio activity with available energy.

5.1 Future Research Directions

In addition to important issues investigated in this dissertation, there are many other interesting and challenging problems for building versatile extremely low duty-cycle sensor networks. For example, for many applications, they have Quality-of-Service (QoS) requirements such as maximum communication delay from a sensor node to a central data gathering base station. How can we achieve such QoS requirements with minimal energy consumption?

As cyber-physical systems expand, every individual person will ultimately become a mobile node within the system. It is estimated that by 2010, there will be a billion transistors per human. For any single person, he/she will connect to and interact with surrounding cyber-physical systems via communication devices with possible sensing capabilities, such as smart phones, smart attire and so on. In order to make those devices that are carried by people as less intrusive as possible and minimize the frequency required for people to recharge those devices, we should start to investigate low duty-cycle designs for mobile systems. When combining low duty-cycle operation and mobility in the network, there are many interesting challenges need to be solved. For

example, in terms of communication, low duty-cycle operation partitioned network in the time dimension while mobility partitioned network in the space dimension, how we can design efficient spatiotemporal solutions for data communication in such networks? Secondly, there are many excellent existing low duty-cycle designs for static systems and mobility solutions for non-energy-sensitive systems. How can we bridge previous knowledge on those two areas and provide effective solutions/middleware to glue those two original orthogonal research areas together?

Bibliography

[1] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A Wireless Sensor Network for Target Detection, Classification, and Tracking. *Computer Networks (Elsevier)*, 2004.

[2] R. Szewczyk, A. Mainwaring, J. Anderson, and D. Culler. An Analysis of a Large Scale Habit Monitoring Application. In *SenSys'04*, 2004.

[3] Zo Abrams, Ashish Goel, and Serge Plotkin. Set K-Cover Algorithms for Energy Efficient Monitoring in Wireless Sensor Networks. In *IEEE IPSN*, 2004.

[4] Mihaela Cardei, My T. Thai, Yingshu Li, and Weili Wu. Energy-Efficient Target Coverage in Wireless Sensor Networks. In *IEEE INFOCOM*, 2005.

[5] S. Kumar, T. H. Lai, and J. Balogh. On k-coverage in a mostly sleeping sensor network. In *Mobicom*, 2004.

[6] S. Slijepcevic and M. Potkonjak. Power Efficient Organization of Wireless Sensor Networks. In *IEEE ICC*, 2001.

[7] D. Tian and N.D. Georganas. A Node Scheduling Scheme for Energy Conservation in Large Wireless Sensor Networks. *Wireless Communications and Mobile Computing Journal*, May 2003.

[8] T. Yan, T. He, and J.A. Stankovic. Differentiated Surveillance Service for Sensor Networks. In *SenSys 2003*, November 2003.

[9] Qing Cao, Tarek Abdelzaher, Tian He, and John Stankovic. Towards Optimal Sleep Scheduling in Sensor Networks for Rare Event Detection . In *IPSN'05*, 2005.

[10] C.-F. Chiasserini and M. Garetto. Modeling the performance of wireless sensor networks. In *INFOCOM 2004: 23th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2004.

[11] W. Choi and S. K. Das. A novel framework for energy - conserving data gathering in wireless sensor networks. In *INFOCOM 2005: 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2005.

[12] Chih-Fan Hsin and Mingyan Liu. Network Coverage Using Low Duty-Cycle Sensors: Random & Coordinated Sleep Algorithms. In *The 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04)*, Berkeley, California, April 2004.

[13] S. Shakkottai, R. Srikant, and N. Shroff. Unreliable sensor grids: coverage, connectivity and diameter. In *INFOCOM 2003: 22th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2003.

[14] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks. In *SenSys'03*, 2003.

[15] H. Zhang and J.C. Hou. Maintaining Sensing Coverage and Connectivity in Large Sensor Networks. *Wireless Ad Hoc and Sensor Networks: An International Journal*, 1, 2005.

[16] Chong Liu, Kui Wu, Yang Xiao, and Bo Sun. Random coverage with guaranteed connectivity: Joint scheduling for wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 17(6), 2006.

[17] Chao Gui and Prasant Mohapatra. Power Conservation and Quality of Surveillance in Target Tracking Sensor Networks. In *MobiCom'04*, 2004.

[18] Shansi Ren, Qun Li, Haining Wang, Xin Chen, and Xiaodong Zhang. Analyzing Object Tracking Quality under Probabilistic Coverage in Sensor Networks. *ACM Mobile Computing and Communications Review*, 9(1), January 2005.

[19] Omprakash Gnawali, Ben Greenstein, Ki-Young Jang, August Joki, Jeongyeup Paek, Marcos Vieira, Deborah Estrin, Ramesh Govindan, and Eddie Kohler. The tenet architecture for tiered sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 153–166, 2006.

[20] Jonathan Hui and David Culler. The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. In *SenSys'04*, 2004.

[21] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In *Proc. of MOBICOM*, August 1999.

[22] C. Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. In *WSNA '02*, 2002.

[23] Y. Yu, B. Krishnamachari, and V.K. Prasanna. Energy-latency tradeoffs for data gathering in wireless sensor networks. *INFOCOM 2004: 23th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2004.

[24] Joengmin Hwang, Yu Gu, Tian He, and Yongdae Kim. Realistic sensing area modeling. In *INFOCOM '07: Proceedings of the 26th Annual IEEE Conference on Computer Communications Mini-Symposiums*, 2007.

[25] V. T. Paschos. A Survey of Approximately Optimal Solutions to Some Covering and Packing Problems. In *ACM Computing Surveys*, June 1997.

[26] Eric W. Weisstein. The reaching algorithm. *http://mathworld.wolfram.com/ReachingAlgorithm.html*.

[27] R. Williams. *Geometrical Foundation of Natural Structure:A Source Book of Design*. Dover Publications Inc, New York, 1979.

[28] F. Ye, G. Zhong, S. Lu, and L. Zhang. PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks. In *ICDCS*, May 2003.

[29] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The Flooding Time Synchronization Protocol. In *SenSys'04*, 2004.

[30] C. Gui and P. Mohapatra. Virtual patrol: a new power conservation design for surveillance using sensor networks. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 33, 2005.

[31] Radu Stoleru, Tian He, John A. Stankovic, and David Luebke. High-Accuarcy, Low-Cost Localization System for Wireless Sensor Networks. In *Third ACM Conference on Embedded Networked Sensor Systems (SenSys 2005)*, November 2005.

[32] Radu Stoleru, Pascal Vicaire, Tian He, and John A. Stankovic. Stardust: a flexible architecture for passive localization in wireless sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, 2006.

[33] N. Patwari, A.O. Hero, M. Perkins, N.S. Correal, and R.J. O'Dea. Relative location estimation in wireless sensor networks. *IEEE Transactions on Signal Processing*, 51(8):2137–2148, 2003.

[34] Santosh Kumar, Ten H. Lai, and Anish Arora. Barrier Coverage With Wireless Sensors. In *MobiCom 2005*, 2005.

[35] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A High Throughput Path Metric for Multi-Hop Wireless Routing. In *MOBICOM'03*, 2003.

[36] Karim Seada, Marco Zuniga, Ahmed Helmy, and Bhaskar Krishnamachari. Energy-efficient Forwarding Strategies for Geographic Routing in Lossy Wireless Sensor Networks. In *SenSys '04*, 2004.

[37] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel. Delay Efficient Sleep Scheduling in WIreless Sensor Networks. In *INFOCOM'05*, 2005.

[38] H.A. Kiehne. *Battery Technology Handbook*. Marcel Dekker, 2003.

[39] Yu Gu, Joengmin Hwang, Tian He, and David Hung-Chang Du. uSense: A Unified Asymmetric Sensing Coverage Architecture for Wireless Sensor Networks. In *ICDCS '07*, 2007.

[40] J. Liu, F. Zhao, P. Cheung, and L. Guibas. Apply Geometric Duality to Energy-efficient Non-local Phenomenon Awareness using Sensor Networks. *IEEE Wireless Communications*, 11(6), 2004.

[41] Y.J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. In *NSDI'05*, 2005.

[42] J. Zhao and R. Govindan. Understanding Packet Delivery Performance In Dense Wireless Sensor Networks. In *Sensys 03*, 2003.

[43] Gang Zhou, Tian He, and John A. Stankovic. Impact of Radio Irregularity on Wireless Sensor Networks. In *MobiSys'04*, June 2004.

[44] Alec Woo, Terence Tong, and David Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *SenSys 2003*, 2003.

[45] SeoungBum Lee, Kyung Joon Kwak, and Andrew T. Campbell. Solicitation-Based Forwarding for Sensor Networks. In *SECON'06*, 2006.

[46] Olivier Dousse, Petteri Mannersalo, and Patrick Thiran. Latency of wireless sensor networks with uncoordinated power saving mechanisms. In *MobiHoc '04*, 2004.

[47] Yang Yu, Bhaskar Krishnamachari, and Viktor K. Prasanna. Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks. In *INFOCOM*, 2004.

[48] Abtin Keshavarzian, Huang Lee, and Lakshmi Venkatraman. Wakeup Scheduling in Wireless Sensor Networks. In *MobiHoc*, 2006.

[49] Joe Polastre and David Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *SenSys'04*, 2004.

[50] Wei Ye, John Heidemann, and Deborah Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *INFOCOM*, 2002.

[51] Yuan Li, Wei Ye, and John Heidemann. Energy and Latency Control in Low Duty Cycle MAC Protocols. In *WCNC'05*, 2005.

[52] Wei Ye, Fabio Silva, and John Heidemann. Ultra-low Duty Cycle MAC with Scheduled Channel Polling. In *SenSys '06*, 2006.

[53] Marco Zuniga and Bhaskar Krishnamachari. Analyzing the Transitional Region in Low Power Wireless Links. In *IEEE SECON'04*, 2004.

[54] Tian He, Sudha Krishnamurthy, Liqian Luo, Ting Yan, Radu Stoleru, Gang Zhou, Qing Cao, Pascal Vicaire, John A. Stankovic, Tarek F. Abdelzaher, Jonathan Hui, and Bruce Krogh. VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance. *ACM Transaction on Sensor Networks*, 2006.

[55] N. Shrivastava, R. Mudumbai U. Madhow, and S. Suri. Target Tracking with Binary Proximity Sensors: Fundamental Limits, Minimal Descriptions, and Algorithms. In *SenSys '06*, 2006.

[56] Ivan Stoianov, Lama Nachman, Samuel Madden, and Timur Tokmouline. PIPENET: A Wireless Sensor Network for Pipeline Monitoring. In *IPSN'07*, 2007.

[57] Chipcon. *CC2420 Product Information and Data Sheet.* Avaiable at http://www.chipcon.com/.

[58] Shan Lin, Jiangbin Zhang, Gang Zhou, Lin Gu, Tian He, and John A. Stankovic. ATPC: Adaptive Transmission Power Control for Wireless Sensor Networks. In *Sensys 06*, 2006.

[59] David Gay, Phil Levis, Rob von Behren, Matt Welsh, Eric Brewer, and David Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *PLDI'03*, 2003.

[60] A. Arora et al. Exscal: Elements of an extreme scale wireless sensor network. In *RTCSA*, 2005.

[61] Jaehoon Jeong, Yu Gu, Tian He, and David Du. VISA: Virtual Scanning Algorithm for Dynamic Protection of Road Networks. In *Proc. of the 28th Annual IEEE Conference on Computer Communications (INFOCOM '09)*, 2009.

[62] H. Morcos, A. Bestavros, and I. Matta. Amorphous placement and informed diffusion for timely field monitoring by autonomous, resource-constrained, mobile sensors. In *SECON*, 2008.

[63] JeongGil Ko, Tia Gao, and Andreas Terzis. Empirical study of a medical sensor application in an urban emergency department. In *BodyNets*, 2009.

[64] Jie Wu, Shuhui Yang, and M. Cardei. On maintaining sensor-actor connectivity in wireless sensor and actor networks. In *INFOCOM*, 2008.

[65] O. Younis, M. Krunz, and S. Ramasubramanian. Coverage without location information. In *ICNP*, 2007.

[66] Yun Mao, Feng Wang, Lili Qiu, Simon Lam, and Jonathan Smith. S4: Small state and small stretch routing protocol for large wireless sensor networks. In *NSDI*, 2007.

[67] Anh Tuan Hoang and Mehul Motani. Collaborative broadcasting and compression in cluster-based wireless sensor networks. *ACM TOSN*, 3(3), 2007.

[68] Marco Zamalloa, Karim Seada, Bhaskar Krishnamachari, and Ahmed Helmy. Efficient geographic routing over lossy links in wireless sensor networks. *ACM TOSN*, 4(3), 2008.

[69] S. Meninger, J.O. Mur-Miranda, R. Amirtharajah, A. Chandrakasan, and J. Lang. Vibration-to-electric Energy Conversion. In *ISLPED*, 1999.

[70] M. Rahimi, H. Shah, G.S. Sukhatme, J. Heidemann, and D. Estrin. Studying the Feasibility of Energy Harvesting in a Mobile Sensor Network. In *ICRA'03*, 2003.

[71] S. Wright, D. Scott, J. Haddow, and M. Rosen. The Upper Limit to Solar Energy Conversion. In *IECEC*, 2000.

[72] Chulsung Park and P.H. Chou. AmbiMax: Autonomous Energy Harvesting Platform for Multi-Supply Wireless Sensor Nodes. In *SECON'06*, 2006.

[73] Aman Kansal, Dunny Potter, and Mani B. Srivastava. Performance Aware Tasking for Environmentally Powered Sensor Networks. In *SIGMETRICS '04*, 2004.

[74] Ting Zhu, Ziguo Zhong, Yu Gu, Tian He, and Zhi-Li Zhang. Leakage-Aware Energy Synchronization for Wireless Sensor Networks. In *MobiSys '09*, 2009.

[75] X. Jiang, J. Polastre and D. Culler. Perpetual Environmentally Powered Sensor Networks. In *IPSN'05*, 2005.

[76] Yu Gu and Tian He. Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links. In *SenSys '07*, 2007.

[77] Yan Wu, Sonia Fahmy, and Ness B. Shroff. Energy Efficient Sleep/Wake Scheduling for Multi-Hop Sensor Networks: Non-Convexity and Approximation Algorithm. In *INFOCOM*, 2007.

[78] Lu Su, Changlei Liu, Hui Song, and Guohong Cao. Routing in intermittently connected sensor networks. In *ICNP*, 2008.

[79] Shibo He, Jiming Chen, David Yau, Huanyu Shao, and Youxian Sun. Energy-efficient capture of stochastic events by global- and local-periodic network coverage. In *MobiHoc*, 2009.

[80] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B. Srivastava. Power Management in Energy Harvesting Sensor Networks. *TECS*, 6(4), 2007.

[81] Christopher Vigorito, Deeak Ganesan, and Andrew Bartoeeee. Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks. In *SECON'07*, 2007.

[82] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler. Trio: Enabling Sustainable and Scalable Outdoor Wireless Sensor Network Deployments. In *IPSN'06*, 2006.

[83] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. *LNCS*, pages 239–254, 2004.

[84] T. Spyropoulos, K. Psounis, and C.S. Raghavendra. Efficient routing in intermittently connected mobile networks: The multiple-copy case. *IEEE/ACM TON*, 16(1):77–90, 2008.

[85] M. Musolesi, S. Hailes, and C. Mascolo. Adaptive routing for intermittently connected mobile ad hoc networks. In *WoWMoM'05*, 2005.

[86] Shuo Guo, Yu Gu, Bo Jiang, and Tian He. Opportunistic Flooding in Low-Duty-Cycle Wireless Sensor Networks with Unreliable Links. In *MobiCom '09*, 2009.

[87] Feng Wang and Jiangchuan Liu. Duty-cycle-aware broadcast in wireless sensor networks. In *INFOCOM'09*, 2009.

[88] Bennett Chow and Dong-Ho Tsai. Geometric Expansion of Convex Plane Curves. In *Journal of Differential Geometry 44 312-330*, 1996.

[89] Pieter C. Allaart and Kiko Kawamura. Extreme Values of some Continuous Nowhere Differentiable Functions. *Math. Proc. Camb. Phil. Soc*, 140(2), 2006.

Appendix A

uSense Appendix

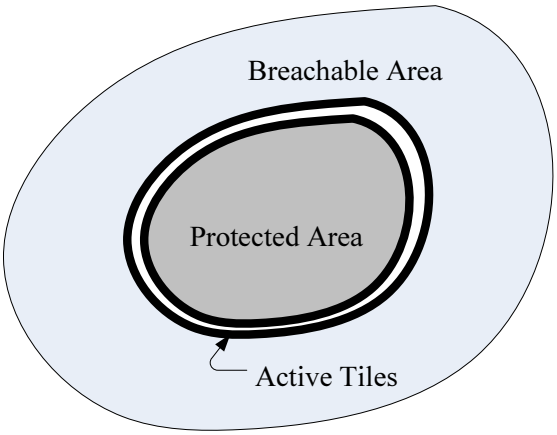


Figure A.1: The Optimality of Systolic Scan

In this appendix, we briefly sketch the proof that systolic scan is the optimal scan when target speed is high. As shown in Figure A.1, in order to avoid complete breach under a high target speed, any coverage algorithm must create a region with a closed curve formed by active tiles *at any time*. We call the area within the closed curve(including the boundary), the *protected area*. In order to avoid a breach, tiles must be turned on consecutively, the active tiles at time $t+1$ must be the neighboring tiles of the active tile at time t. Clearly, the more tiles turned on at each time interval, the smaller WCB will be. Given a closed curve, the maximum number of tiles turns on at time $t+1$ can be achieved through geometric expansion [88], which is the operation defined by systolic

scan. We note that when the shape of the area is irregular (especially concave), the formal proof of optimality is non-trivial and out of the scope of this dissertation. More information can be found at [88].

Appendix B

ESC Appendix

In this section, we prove the optimality for solving multiple active instances augmentation or decrement with greedy selections. Essentially, we need to show both greedy choice and optimal substructure properties. For greedy choice property, it is obvious since either single active instance augmentation or decrement guarantees the minimal cross-traffic delay at a node when the node duty-cycle increases or decreases.

To prove the optimal substructure property, it is sufficient to prove that by augmenting or decreasing an active instance to the optimal solution of augmenting or decreasing $n-1$ active instances, we can obtain the optimal solution for augmenting or decreasing n active instances at a node. According to Lemma 4.5.1, the cross-traffic delay at a node only depends on the number of active instances in each of time intervals that are partitioned by active instances of predecessors and successors. Assume there are c time intervals, the cross-traffic delay at a node b for a packet ready time t at a predecessor a therefore can be represented in the form of $D_a(t) = f(x_1^t, x_2^t, x_3^t .., x_c^t)$, where x_i^t is the number of active instances within the i^{th} time interval after time t, $i = 1, 2, ..., c$.

For example, assume node c is the successor of node b and it has an active instance at t_1^c, the number of active instances within interval (t, t_1^c) and (t_1^c, t) therefore is x_1^t and x_2^t, respectively. The cross-traffic delay within interval (t, t_1^c) is:

$$\sum_{i=1}^{x_1^t} P^{ab}(i)(t_i^b - t + t_1^c - t_i^b) = (t_1^c - t)[1 - (1 - p_{ab})^{x_1^t}]$$

Similarly, the cross-traffic delay for the interval (t_1^c, t) is:

$$\sum_{i=x_1^t+1}^{x_1^t+x_2^t} P^{ab}(i)(t_i^b - t + t_1^c + T_r - t_i^b) = (t_1^c + T_r - t)[1 - (1 - p_{ab})^{x_2^t}](1 - p_{ab})^{x_1^t}$$

Adding delays in two intervals together, we have:

$$D_a(t) = (t_1^c - t)(1 - p_{ab}^{x_1^t}) + (t_1^c + T_r - t)(1 - p_{ab}^{x_2^t})p_{ab}^{x_1^t}$$
$$= [t_1^c - t - (t_1^c + T_r - t)(1 - p_{ab})^{N_c}] + T_r(1 - p_{ab})^{x_1^t}$$

Formally, let $q = 1 - p_{ab}$, $D_a(t)$ can be written in the form of exponential sum:

$$D_a(t) = a_0 + a_1 q^{x_1^t} + a_2 q^{x_1^t + x_2^t} + ... + a_{c-1} q^{x_1^t + x_2^t + ... + x_{c-1}^t} \tag{B.1}$$

where a_i/a_0 is a constant value and $a_1 < a_2 < ... < a_{c-1}$, $i = 0, 1, 2, .., c-1$.

Cross-traffic at node b (D_b), is linear combination of Equation B.1 for all packet ready times at all predecessor nodes of node b (Equation B.1). Let $x_1, x_2, ..., x_c$ be the number of active instances within the i^{th} time interval after the beginning of a period, D_b then is a function of $x_1, x_2, ..., x_c$. Let $A_i(x_1, x_2, ..., x_i, ..., x_c)$ be the terms that include x_i in D_b, then the problem of augmenting n active instances can be solved by repeating following procedure n times.

1. For each interval i, we attempt to augment an active instance and record the value of reductions for $D - b$. Formally, we can represent the decrease of D_b by augmenting an active instance in time interval i as $\Delta A_i = A_i(x_1, x_2, ..., x_i, ..., x_c) - A_i(x_1, x_2, ..., x_i + 1, ..., x_c)$.

2. Let $\Delta A_k = Max(\Delta A_i)$, where $i = 1, 2, ..., c$, we increase x_k by one.

Since in each of above iteration, we reduce D_b by a maximum value, above process guarantees the optimality of n active instances augmentation [89]. Consequently, we prove the optimal substructure property for multiple active instances augmentation. Similarly, we can prove the case for multiple active instances decrement.