ENERGY-EFFICIENT INFORMATION PROCESSING AND ROUTING IN WIRELESS SENSOR NETWORKS — CROSS-LAYER OPTIMIZATION AND TRADEOFFS

by

Yang Yu

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

August 2005

Dedication

To my dear parents and sister!

Thank you all for your love!

Acknowledgements

First and foremost, I would like to express my deepest appreciation and gratefulness to my advisors, Prof. Prasanna and Prof. Krishnamachari. They have not only taught me knowledge, skills, and experience in terms of research, but also mentored me in my life and career. I thank them for all their guidance, support, inspiration, and encouragement during the past years.

I would like to thank other members on my defense and qualify exam committee, including Prof. Suhkatme, Prof. Raghavendra, Prof. Pedram, and Prof. Golubchik. They have offered many useful feedback that finally brings the completion of this thesis.

During these years, I have been working in both Prof. Prasanna's research group — P-Group and Prof. Krishnamachari's group — Autonomous Network Research Group (ANRG). The pleasant research atmosphere in both groups is so helpful and enjoyable. I would like to thank all the members in the group, including Ammar Alhusaini, Zachary Baker, Amol Bakshi, Seonil Choi, Sethavidh Gertphol, Gokul Govindu, Bo Hong, Shyam Kapadia, Gang Lu, Sumit Mohanty, Gerald R. "Jerry" Morris, Jingzhao Ou, Neungsoo Park, Animesh Pathak, Sundeep Pattem, Narayanan Sadagopan, Ronald Scrofano, Reetinder Sidhu, Mitali Singh, Dongjin Son, Thrasyvoulos Spyropoulos (Akis), Avinash Sridharan, Rahul Urgaonkar, Kiran Yedavalli, Cong Zhang, Ling Zhuo, Marco Zuniga. Of these, I want to give special thanks to Bo, who has been my roommate for one year; Mitali and Akis, who have been sharing office with me for most of the time; Gang and Nara, whom I have discussed lots of research problems with; and Jingzhao, who is a very good and helpful friend in life.

I also want to take the opportunity to thank Prof. Siegel, Prof. Maciejewski, Prof. Ali, and Jong-Kook Kim for their collaboration and help during the first two years of my Ph.D. study.

I want to express my wholehearted gratitude to my parents and other family members. They have supported me so much during all these five years, which is a long and rough journey for both me and them. I own them too much!

Finally, I want to thank my close friends, including Fan Bai, Tao Ding, Xuhua Ding, Juan Huang, Ji Lin, Qiang Ni, Yangbing Wu, Hongwei Wu, Ning Xu, Weiqing Xu, Junhao Yang, Sheng Yang, Yan Zhou, and Mingrui Zhu for their cherished friendship and all the happiness they have brought to me. I really appreciate Fan, Xuhua, and Sheng for being my roommates among these years and being so helpful in both my life and research.

Contents

List Of Tables

List Of Figures

Abstract

Wireless sensor networks have become an important technology to realize many applications, including both simple event/phenomena monitoring applications and heavy-duty data streaming applications. While many systems are being developed, we focus on two fundamental operations: information processing and information routing. These two operations are tightly related and must be performed in a collaborative fashion.

A major concern in designing and operating sensor networks is their energy-efficiency. Cross-layer optimization is widely accepted as an effective technique to ameliorate this concern. The basic idea is to share information across different system layers and to enable tradeoffs involving multiple layers, which provides a larger optimization space for system design. It is natural to apply cross-layer optimization techniques in the context of collaborative information processing and routing. In this thesis, we investigate three specific problems in this emerging and important research area.

The first research effort addresses **collaborative data processing** in a single hop cluster that behaves as a basic operating unit across the network. We investigate the assignment and scheduling of a set of real-time communicating tasks onto the cluster under a novel performance metric — to balance the energy cost of all nodes within the cluster. We focus on exploring the energy-latency tradeoffs with adjustable computation and communication speed, enabled by techniques such as voltage scaling and rate adaptation. We have developed integer linear programming formulations for optimal solutions as well as a 3-phase heuristic. We have achieved up to 10x lifetime improvements in simulated scenarios.

The second research effort considers the **transportation of information** to the base station over an existing routing substrate (i.e., a data gathering tree) within a user-specified latency constraint. We again explore the energy-latency tradeoffs through rate adaptation. By

exploiting the dependency between communication links over the tree, we have developed both off-line and on-line techniques to adjust the communication speed for energy minimization. Energy conservation up to 90% is achieved by our techniques.

The third research effort investigates the **construction of a routing tree** that minimizes the total energy costs of data compression and communication. Such an objective is novel compared with traditional maximum compression philosophy and is crucial for advanced computation-intensive applications where a balance between computation and communication energy is necessary. We utilize the concept of tunable compression with a suitable model to capture the tradeoffs between the compressing time and the output size. By revealing the inherent tradeoffs between two simple tree constructions — shortest path tree and minimal steiner tree — via both analysis and simulation, we show that the minimal steiner tree is a practical solution with acceptable performance for systems with both grid and arbitrary deployment.

The three research thrusts were chosen to cover various operating stages of information processing and routing in sensor networks. They provide a framework above which various extensions can be overlaid.

While our work has made significant contributions to the field, we also discuss advancements to our work that will further increase the capabilities of collaborative information processing and routing in sensor networks. Specifically, we identify a set of future research consideration for adaptive fidelity algorithms, node mobility, unreliable wireless communication, and integration with existing technologies.

Chapter 1

Introduction

In this chapter, we provide an introduction of this thesis by giving the general background, our research motivation and goal, and a brief description of our approaches and contributions.

1.1 Overview

With the advancements in Micro-Electro-Mechanical Systems (MEMS) and miniaturization techniques, wireless sensor networks (WSNs) emerged as a new technology and research topic around six years ago [40, 54, 88, 82]. Such systems usually consist of a large number of tiny and cheap sensor nodes that are capable of autonomous sensing, computing, communication, or even actuation. After the initial deployment (typically in an ad hoc manner), these sensor nodes are responsible for self-organizing an appropriate network infrastructure over which information sensing, processing, and transportation can be performed. In typical scenarios, users can retrieve information of interest from the network by injecting queries and gathering results from the so-called base stations (or sink nodes) which behave as an interface between users and the network.

WSNs are envisioned to be used in various contexts of applications, including environment monitoring, habitat study, battlefield surveillance, infrastructure monitoring. Some of the applications have been previously studied using more complex, more powerful, and also more expensive sensing equipments, such as remote battlefield surveillance. The development of WSNs enables a new solution domain for such applications that has several unique advantages, including easier and faster deployment, cheaper cost, closer sensing distance and hence potentially higher signal to noise ratio (SNR), information fusion based on much more sensing samples, and fault tolerance through high redundancy of sensor nodes. The philosophy is that while the capability of each individual sensor node is quite limited, the aggregated power of the entire network is sufficient for the required mission.

The development of WSNs is accompanied by a series of research challenges. In the following, we list a few that are of particular interest to the work in this thesis.

First of all, the operating paradigm of WSNs is centered around information retrieval from the underlying network, usually referred to as a *data-centric* paradigm. Compared to the *address-centric* paradigm exhibited by traditional networks, data-centric paradigm is unique in several ways, including a new communication pattern that resembles a reverse multicast tree (usually referred to as a data gathering tree), in-network processing to extract information from raw data and remove redundancy among data from multiple sources, and cooperative strategies among sensor nodes instead of Internet-based non-cooperative strategies. The development of appropriate routing strategies that take the above factors into consideration is challenging. We will discuss in detail about data-centric paradigm in Section 1.2.

Second, the above data-centric paradigm involves two fundamental operations in WSNs, namely information processing and information routing. A large portion of work in this thesis is motivated by the fact that information processing and routing are mutually beneficial in the sense that while information processing helps reduce data volume to be routed, information

routing facilitates joint information compression (or data aggregation) by bringing together data from multiple sources. However, it is often non-trivial to model and analyze the inter-relationship between information processing and routing. In many situations, the problem of finding a routing scheme cooperated with joint compression for energy minimization turns out to be NP-hard. We will further elaborate this issue in Section 1.3.

Third, since once deployed it is often infeasible or undesired to re-charge sensor nodes or replace their batteries, energy conservation becomes crucial for sustaining a sufficiently long network lifetime. Among the various techniques that have been proposed for improving energy-efficiency of the system, cross-layer optimization has been realized as an effective approach. Application specific design of WSNs makes it unnecessary to develop strictly layered structure to support application generality, compared to the layering overhead. More importantly, due to the nature of wireless communication, one performance metric of the network can be affected by various factors across layers. Also, one optimization technique applied at a specific layer often affects the behavior or performance metrics at other layers. Hence, a holistic approach that simultaneously considers the optimization at multiple layers enables a larger design space within which cross-layer tradeoffs can be effectively explored. We discuss issues related to cross-layer optimization in Section 1.4.

It is natural to applied cross-layer optimization techniques in the context of information processing and routing, which is the main theme of this thesis. We summarize a list of techniques, referred to as *system knobs*, that can be applied for this purpose in Section 1.5. Out of these techniques, we are particularly interested in three of them, namely voltage scaling, rate adaptation, and tunable compression. These techniques address the issue of energy conservation from computation, communication, and joint compression perspectives, respectively.

Specifically, voltage scaling and rate adaptation achieve energy savings by trading computation/communication delay for energy, while tunable compression explores the tradeoffs between computation and communication energy costs.

Based on the above cross-layer optimization techniques, we study three specific topics in the context of information processing and routing in WSNs. These three topics are information processing within a cluster, information transportation over a tree substrate, and joint information routing with tunable compression. These topics are originated from either simple environment monitoring applications with small data volume, or advanced computation-intensive applications such as video surveillance. While voltage scaling and rate adaptation have been widely studied in literature, their application in the context of information processing and routing in WSNs has not been previously addressed. To the best of our knowledge, this is also the first work to formally consider tunable compression with routing in WSNs. Detailed description about the three topics and the contributions of this thesis are presented in Section 1.6.

1.2 Data-Centric Paradigm

The set of applications that are envisioned for WSNs include environment monitoring, habitat study, battlefield surveillance, infrastructure monitoring. All of these applications require the sensing, processing, and gathering of information from the physical environments where the network operates. The end users are interested in the content of the information, including related spatial and temporal specification. Hence, the computation and communication are centered around the information itself, instead of the sensor nodes that sense or hold the information. This feature is called *data-centric* computation and communication paradigm [68], as opposite to the address-centric or node-centric paradigm, which is exhibited in traditional Internet networks, mobile ad hoc networks, or parallel and distributed systems.

The difference between data-centric paradigm and address-centric paradigm can be effectively reflected by the user queries. For example, typical user queries or service requests in address-centric paradigm are "Load the web page at address ...", "Transfer file A from host B to host C", etc. However, in data-centric paradigm systems, typical queries are "What is the average temperature at region A within time T?", "Are there any vehicles currently in region A?". For these queries, the users are interested purely in the information itself, but not the sensor nodes that generate or hold the information, neither the way how the information is transmitted to the end users.

From a more formal perspective, data-centric paradigm differs from address-centric paradigm in the following aspects. First, the typical goal of WSNs with data-centric paradigm is to gather specific information from multiples source nodes to a small number of sink nodes. Hence, the communication pattern normally resembles *a reverse multicast tree*, instead of a more randomized peer-to-peer-based communication pattern in address-centric systems. The works in this thesis are largely based on this tree structure.

Second, most of data gathered from a physical environment are not of direct interest to the end users and they often show strong correlation. Therefore, it is necessary to process the data before they are transported to the end users. Such *in-network processing* includes signal and image processing to extract useful information out of the raw data, data compression to reduce communication load, and joint compression of data from multiple sources to eliminate redundancy among the data. These three forms of in-network processing are addressed by this thesis via the aforementioned three specific topics, which will also be detailed later.

Third, the routing schemes that were originally developed for address-centric systems is based on fulfilling each individual data routing request. Hence, they become unsuitable for

data-centric systems, where the communication request of the whole system is application-specific which may be as simple as to route data from all source nodes to a sink node. To investigate new routing schemes that are customized for WSN applications becomes important.

Fourth, since applications of a WSN are usually more specific and more well-defined, the sensor nodes within the system are more likely to work in a *collaborative* fashion, instead of a competing fashion. In this way, the aggregated resource of the system can be more efficiently utilized.

With the above features, data-centric paradigm is the main motivation of the work presented in this thesis. It justifies the use of a tree structure for transporting gathered data. It also raises the necessity of in-network processing, including the processing of raw data into useful information as well as joint data compression integrated with routing along the tree structure. Our work is based on these two important facts.

We note that data-centric paradigm does not mean that address of nodes is not necessary any more. Instead, we will assume the availability of local address in many cases, including both the routing over a tree and in-network processing within a local cluster. Such a local address is necessary for applications that perform peer-to-peer communication beyond simple flooding. Also, our work is not limited to data-centric paradigm only. The techniques presented in this thesis can be adopted for energy-efficient computation and communication in general wireless networks as well.

1.3 Collaborative Information Processing and Routing

In data-centric paradigm, the system operations are centered around the fundamental functionality: to deliver the required information to the end users. Apparently, this involves two operations:

1. *information processing* that includes sensing the environment and extracting useful information out of the raw data, and

2. *information routing* that includes combining the information from difference sources across the network and routing the final set of information to the end users.

Note that in some sense these two operations are actually applicable to address-centric paradigm. For example, users may want to extract information from a file on machine A and then transmit the result to machine B. However, in data-centric paradigm, these two operations are not sequentially and independently performed as they usually are in address-centric paradigm. On the contrary, these two operations are performed in a parallel fashion and are tightly related to each other.

On one hand, information processing can help reduce the data volume to route by extracting useful information from the raw data, or compressing the data, or removing redundancy among data through joint source compression. On the other hand, since source data are distributed across the network, many information processing techniques (especially joint source compression) rely on the data availability at sensor nodes that heavily depends on the routing scheme. Such a mutually beneficial relationship leads to tightly coupled design and implementation of information processing and routing, referred to as *collaborative information processing and routing*.

For example, consider the simple network scheme in Figure 1.1, where vertices A through D denote sensor nodes and edges denote valid communication links. In this example, there are two source nodes, A and B, and one sink node, S. Our mission requires to gather and transport information from both A and B to the sink node. While the route from node B to S has exactly one option which is BCS, there are two alternative routes from node A to S: ACS and ADS. If the two pieces of source information at nodes A and B are relatively independent, the routing selection for A is not crucial in terms of the total communication cost. However, if the source

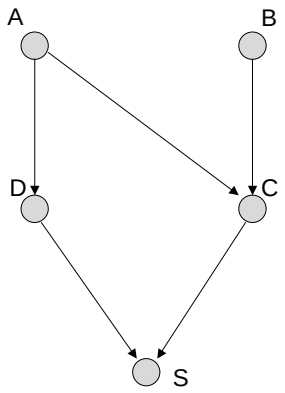


Figure 1.1: An example network scheme for collaborative information processing and routing

information is highly correlated, we can take advantage of joint compression to reduce the data volume to be routed. Hence, the route ACS will be preferred to enable joint compression at node C, given that the gain of reduced data volume to communicate over CS dominates the extra communication cost of choosing ACS instead of ADS, if any. This example clearly indicates the importance of coupling the selection of routing schemes with data compression techniques.

Since sensor readings from the physical world are usually highly correlated, the above joint compression technique is particularly useful to reduce data volume by removing redundancy among data from multiple sources. Such a joint compression is also referred to as *data aggregation* [68]. We will use the terms data aggregation and joint compression exchangeably hereafter.

Choosing an appropriate routing scheme with joint compression in general networks is much more challenging, since it is affected by many factors, including network topology, communication cost and reliability, computation cost, and the correlation among data. In many general settings, to choose the optimal routing scheme is NP-hard and hence heuristic solutions are

preferred. For example, consider the problem with a simplified assumption that a perfect data aggregation can be achieved. This means the output of jointly compressing data from an arbitrary number of sources is exactly one unit data. To form a routing scheme from a given set of source nodes to a sink node under this assumption to minimize the total communication cost in terms of bit times hops is the Minimal Steiner Tree problem, which is known to be NP-hard[1].

In many real-life cases without perfect data aggregation, to choose an appropriate model for abstracting the data aggregation operation is even more difficult, since it heavily depends on the data correlation from multiple sources. Such a data correlation is determined by the nature of the observed phenomena as well as the physical situation of the operating field which affects the propagation of the phenomena. While several papers have proposed models to abstract the correlation among data from the physical world [97, 78, 87], a unified and widely accepted model is still lacking. Hence, several research efforts have been trying to study the inter-relationship between information processing and routing under certain simplifications. For example, the work by Cristescu *et al.* [32] assumes a fixed reduction in data volume for any source data which is jointly compressed by data from other sources. Some other models are discussed in Section 1.5.4.

Another important research direction in order to cope with the above challenge is to perform joint data compression from information coding perspective. For example, the Slepian-Wolf coding [111] can be used to code two correlated sources with a total data volume equal to the joint entropy of the two sources without explicit communication between the two sources. Indeed, the Slepian-Wolf coding is studied in [32], which shows that the routing selection and joint data compression can be perfectly de-coupled with such a coding scheme. However, such a theoretical limit has not been achieved by any practical distributed source coding schemes yet. Hence, we focus on joint data compression with explicit communication.

[1]Please note that the routing scheme does not necessarily need to be a tree in general cases. However, such a tree structure is the focus of many studies due to its simplicity.

Most of the existing works on information processing and routing are intended for a single sink node. When multiple sink nodes are considered, the concept of *network information flow* [2] can be used to design a joint routing and coding scheme that transports required information to all sources under a given network capacity. However, joint data compression is not considered in the original definition of network information flow. How to integrate these two concepts in the context of WSN applications is an open challenge.

Moreover, most research efforts in literature assume a static system model in terms of communication reliability, network topology, and data correlation. The problem becomes even more challenging if we consider the possibly high dynamics in the above system parameters in real-life scenarios.

1.4 Cross-Layer Optimization for Energy-Efficiency

1.4.1 Motivation

A major concern for WSNs is energy-efficiency, which has been addressed by various techniques targeting hardware components [54, 90], media access layer (MAC) scheduling policies [128, 75], network organization [52, 106], routing protocols [61, 45], signal processing techniques [5, 13], and application level algorithms [109, 107]. It has also been realized that cross-layer optimization is of particular importance for saving energy in WSNs, since it enables a large space for tradeoffs and optimization of performance metrics across different layers [93, 103, 134].

Network layers (or stacks) are designed for traditional networks with the goal to decompose a complex system into several layers with well-defined interface among them so that the modification within each layer only needs to respect the pre-defined interfaces with other layers. Hence, each layer behaves like a black box from the perspective of other layers. However, such a clean abstract is useful for designing complex systems with emphasis on functional generality

at the expense of decreased efficiency. When system complexity or functional generality is not a crucial concern, awareness of the implementation within each layer enables more efficient designs. The comparison between general purpose CPU and application-specific integrated circuit (ASIC) is a good example.

The above notion of cross-layer optimization is the main focus of this thesis. Although cross-layer optimization has been widely studied in many contexts, there are unfortunately no formal definitions for cross-layer optimization. From a broad perspective, we present our definition as follows:

any hardware/software techniques applied at a specific system layer can be regarded as cross-layer optimization if they explicitly interact with the functionalities or optimization techniques at other system layers and in most cases, explicitly affect the system properties or performance at those layers.

The motivation for cross-layer optimization for energy-efficiency is multi-fold. First, system performance metrics in WSNs are often determined by multiple factors across several layers. For example, the performance of wireless communication (either throughput or energy) is jointly determined by several factors across physical, MAC, and routing layers, which is significantly different from wired communication. Also, an efficient routing scheme should take wireless communication conditions, network topology and connectivity, possibility of joint data compression, and application-level quality-of-service requirements into consideration. The optimization within each individual layer often leads to inefficient solutions. Hence, a holistic approach that simultaneously considers different layers with cross-layer information sharing and coordinated optimization enables a much larger design and optimization space. Many research efforts on cross-layer optimization are based on this important hypothesis.

Second, optimization techniques applied at one particular layer often affect the behavior and performance metrics at other layers. For example, sleep scheduling can affect signal interference

at physical layer, channel access at MAC layer, routing selection at routing layer, and sensing coverage at application layer. Isolating optimization techniques at each individual layer may cause conflicts in optimization goals or even counteracting solutions. It is therefor crucial to share information across the system stack and expose the effects of various optimization techniques to all layers so that coordinated optimization can be performed. This factor is however ignored by many research efforts, due to the inherent complexity to cope with multiple performance metrics at multiple layers.

Third, WSNs are usually application specific in terms of the required functionality. The generality of functionalities supported by strictly layered network/system structure becomes unnecessary compared with the layering overhead. Hence, a blurred boundary between layers or even the removal of unused layers helps build a lightweight and more efficient system.

Of course, the advantage of cross-layer optimization is not free. The large design and optimization space also leads to more challenging algorithm and system design and more complicated interactions across various layers. However, given the application specific property of WSNs and fairly limited functionality and capability of sensor nodes, these challenges are expected by most researchers to be tractable and worthwhile. Also, cross-layer optimization does not mean that layering is completely unnecessary. Instead, we still need a layered structure so that a clean abstract is presented at each layer which abstracts unnecessary implementation details from other layers. The key point in cross-layer optimization is the information sharing and coordinated optimization across the stack.

A common way to realize cross-layer optimization is to adjust the system performance by exposing system knobs across layers as well as their impact on the system performance. System knobs are actually tunable parameters that have a significant impact on the performance of the system. One well-known example is the shutdown or sleeping policy that tunes the awake time of the sensor nodes to adjust various upper-layer functionalities, including MAC layer

scheduling [128, 75], topology control [26, 125], routing selection [124], and coverage for event detection [115, 1]. Besides the motivation to reduce channel contention and to alleviate the scalability issue by keeping as a small number of awake sensor nodes as possible, the major principle in this case is to deliver "just enough" performance as required with a minimized resource usage, including energy. Other commonly used system knobs include voltage scaling that adjusts CPU computation speed [127], power control that adjusts radio transmission radius [95], and rate adaptation that adjusts radio transmission speed [89]. We will discuss these system knobs in detail in Section 1.5.

1.4.2 Consideration for Collaborative Information Processing and Routing

It is quite natural to apply cross-layer optimization techniques into the context of collaborative information processing and routing. From one perspective, such techniques can be applied based on the following three operating stages:

1. data sensing at source nodes,
2. signal processing at source nodes, and
3. joint information routing and compression across the network.

We note that besides the last stage, the first two stages also require distributed and coordinated operations among sensor nodes in many cases. Although data sensing seems to be a localized operation that involves each sensor node as a basic function unit, the challenge lies in the fact that the aggregated sensing behavior of all sensor nodes usually needs to satisfy certain coverage requirement [115, 1]. Since the sensing and computation capability of each sensor node is limited, signal processing usually requires the coordination a small group of sensor nodes in proximity to extract useful information from the raw data gathered by the sensor nodes, e.g,

the beamforming algorithm [13]. Hence, for all the three operating stages, the cross-layer optimization is expected to be performed in a distributed fashion.

From another perspective, cross-layer optimization techniques can be classified based on the layer where the techniques are applied. For our purpose, we divide the system into 5 layers: hardware layer, physical layer, multiple access control (MAC) layer, routing layer, application layer. Many cross-layer optimization techniques have been proposed at each of these layers with the general goal of improving the system energy-efficiency. In Table 1.1, we re-interpret them in the context of the above three operating stages of collaborative information processing and routing.

Table 1.1: Examples of cross-layer optimization techniques for energy-efficient collaborative information processing and routing

System layer	Data sensing	Signal processing	Joint information routing and compression
Application		Energy-efficient signal processing, adaptive fidelity algorithms	Joint routing and coding, tunable compression
Routing			Energy-aware routing, entropy-aware routing
MAC			Radio sleep scheduling
Physical			Power control, rate adaptation, adaptive coding
Hardware	Low-power CPU, node sleep scheduling, voltage scaling		Low-power radio

In the next section, we give a brief survey of the system knobs listed in Table 1.1.

1.5 A Brief Survey of Cross-Layer Optimization for Energy-Efficiency Collaborative Information Processing and Routing

We present this brief survey based on the system layers where the techniques are applied, including hardware layer, physical layer, MAC layer, networking layer, and application layer. Note that although we have narrowed down our attention to only optimization techniques for energy-efficiency, this survey is by no means a complete list. Instead, we focus on the techniques that are listed in Table 1.1.

1.5.1 Hardware Layer

The most important hardware layer technique is low-power circuit design for both CPU and radio modules, which has been addressed by a large body of literature [82, 90]. Such low-power design gains energy-efficiency by (1) developing dedicated low-power, low cost hardware modules for the expected low performance of sensor nodes, and (2) exploring tradeoffs between power consumption of the system and other performance metrics. The performance criteria for typical sensor nodes are around tens of MIPS for CPUs (e.g., 16 MIPS for the ATMEL ATmega128L processor [8] used in Berkeley Motes) and tens of Kbps for the radio modules, which are fairly low compared to the performance of usual PCs with commercial wireless LAN cards. This application-level requirement provides opportunities to design simple digital circuits, including digital signal processors (DSPs) and radio frequency (RF) circuits, with less power consumption.

We now discuss two important tradeoffs that have been explored at the hardware layer — the energy *vs* response time tradeoff and the energy *vs* delay tradeoff. The energy *vs* response time tradeoff is realized via shutting down the node in idle state [112, 108], which is motivated

by the well-known ACPI (Advanced Configuration and Power Interface) industry specification. Note that here we discuss the shutdown of the whole node; MAC layer radio sleep scheduling will be discussed in Section 1.5.3. The key points of the tradeoff lie in two aspects: (1) to select the appropriate shutdown mode based on the tradeoffs between shutdown duration and waking-up time/energy cost, and (2) to determine the shutdown duration by exploring the tradeoffs between energy efficiency and possible event miss rate at application level. By converting the temporal event miss rate to the spatial percentage of coverage, the second tradeoff is also studied as the energy *vs* sensing coverage tradeoff [115, 1].

Instead of reducing the operating duration of CPU by the shutdown technique, voltage scaling explores the energy *vs* delay tradeoff by running CPU at a lowered speed and hence longer operating duration with reduced supply voltage and operating frequency [127]. The key rationale is that the CPU power consumption is quadratically proportional to the supply voltage with delay being linearly inverse-proportional to the supply voltage, implying that the power-delay product increases with the supply voltage. Apparently, the energy *vs* delay tradeoff is meaningful for tasks with application-level real-time constraints, which are usually captured by setting a hard or soft deadline for each task. Various research efforts have proposed scheduling techniques for voltage scaling in uni-processor systems [127, 57, 104, 10] or multi-processor systems [49, 138, 76, 135, 132].

A key question regarding the usefulness of the above CPU related techniques is the relative energy cost spent by CPUs compared to that of radio modules. One fact is that the energy cost to transmit one bit is typically around 500 - 1000 times greater than a single 32-bit computation [105, 12]. Hence, for applications with simple functionality, striving for CPU energy-efficiency might not be worthwhile. However, we also envision the development of more advanced, computation-intensive applications within the near future. Moreover, it has been noted that for many high-end sensor nodes, the power consumption of CPU is around 30-50%

of the total power consumption of the system [93], which also motivates energy minimization for CPUs with complex applications.

1.5.2 Physical Layer

At the physical layer, energy-efficiency is often optimized using techniques such as power control, rate adaptation, and adaptive coding that have a direct impact on the signal strength and interference at receivers[2]. Such an impact eventually affects the network connectivity, topology, data transmission rate, and energy costs at various layers, including MAC, networking, and application layers. These complicated cross-layer effects make it difficult to isolate the tradeoffs involved with these techniques. Nevertheless, we can make a first-order classification that power control explores the tradeoff of energy *vs* connectivity and reliability, while rate adaptation and adaptive coding explore the tradeoff between energy and communication speed, or equivalently transmission delay.

The rationale behind these two tradeoffs can be explained using Shannon's law in wireless communication [31]. Consider an Additive White Gaussian Noise channel. This law states that the achievable communication rate is logarithmically proportional to the power at the receiver, which in turn is proportional to the transmission power at the receiver and decays with the transmission distance at a rate of d^{α}, where d is the radius and α is a the pass loss exponent between 2 to 6. Hence, increment of either the communication radius or rate leads to increased transmission power. Following the principle of delivering just enough performance, we would like to decrease power while maintaining just enough communication radius and data rate.

Power control was originally proposed for single-hop multi-user systems like the cellular system to maintain a given level of signal-to-noise quality to compensate for fading effects, thermal noise, and more importantly, mutual interference in the shared radio spectrum [60, 51].

[2]Power control is sometimes treated as a MAC layer technique since it affects the MAC layer topology

In the context of WSNs where multi-hop communication prevails, power control has been mainly used for determining an appropriate communication radius, which can be either common for all sensor nodes [85] or not [95]. Many research efforts have proposed power control schemes to reduce the communication radius and hence the power consumption while achieving global network connectivity [95, 94, 16, 85, 121, 70]. Also, the tradeoff between energy and reliability through power control is studied [69].

Rate adaptation (sometimes also referred to as modulation scaling [98]) and adaptive coding were also originally proposed for cellular systems or local wireless networks with the goal of throughput optimization [117, 55]. The use of these techniques for scheduling packet transmission over a given channel with the goal of minimizing energy cost subject to delay constraint is studied in [89], with an optimal off-line algorithm similar to the one in [127]. The problem is then extended to a star structure with multiple downstream links [43].

Recently, many research efforts are trying to analyze and utilize the joint impact of these two techniques on energy-efficiency and throughput optimization, and many times together with MAC layer transmission scheduling and networking layer routing selection [69, 33, 20, 37, 67]. As it has been realized that energy-efficiency depends on factors spanning multiple layers, this is becoming a promising and hot research direction.

1.5.3 MAC Layer

The way that MAC layer affects the energy-efficiency is mainly through the adjustment of transmission scheduling and channel access. A common way to do that is via sleep scheduling [91, 128, 118, 74] from long time scale perspective or time-division multiple access (TDMA) [6, 75] from short time scale perspective. Similar to the shutdown technique of

CPUs, sleep scheduling also explores the energy *vs* response time tradeoffs in wireless communication. During many studies, the response time is translated to network or application layer transmission delay or throughput.

The PAMAS (Power Aware Multi-Access Protocol with Signaling) protocol [91] is a simple incremental over the MACA protocol [64] by turning off the radios of nodes that cannot either transmit or receive given the current traffic in neighborhood. A more aggressive policy, S-MAC is proposed by Ye *et al.* [128], in which nodes determine their own sleep scheduling based on the sleep scheduling of neighboring nodes. To cope with the problem that the scheduling is pre-determined in S-MAC, a more dynamic policy, T-MAC is proposed so that the scheduling of a node can be adapted on-the-fly based on transmission requests from neighbors [118]. While the above works are proposed for a general network topology, a scheduling policy dedicated to routing tree structure in WSNs is proposed in D-MAC [74]. The main advantage of D-MAC is that it facilitates a fully pipelined packet transmission over the routing tree by staggering the sleep scheduling of nodes.

Compared to the above adaptive sleep scheduling, TDMA provides a more strict, pre-specified sleep scheduling. Tradeoffs between energy and delay as well as buffering size are studied in [6]. A novel performance metric of network diameter is studied by Lu *et al.* [75].

Another tradeoff explored by sleep scheduling is the energy *vs* topology tradeoffs, which in turn impacts concurrent transmission scheduling and channel access at MAC layer and routing selection at networking layer. Most of the research efforts along this line try to maintain a backbone style topology of the network such that the network remains connected with a minimum number of awake sensor nodes [26, 125, 100].

The Span protocol [26] uses a randomize method to elect so-called coordinator nodes to maintain a backbone connectivity with certain redundancy. The concept of virtual grid is proposed in GAF [125], the size of which is determined by the communication radius so that

any nodes in two adjacent virtual grids can communicate with each other directly. The key point is then to ensure exactly one active node in every virtual grid. In the STEM protocol [100], radios are proactivated by using either a paging channel with fixed duty cycle or a tone on a secondary channel, which provides extra means to explore the energy *vs* latency tradeoff.

1.5.4 Routing Layer

The first batch of routing protocols that were adopted for WSNs are mostly based on protocols that were originally proposed for ad hoc networks, including extensions of AODV and DSR [125]. These routing protocols are still based on traditional address-centric peer-to-peer communication patterns instead of data-centric paradigm of WSNs.

Directed diffusion [61] is almost the first well-known protocol customized for information routing in data-centric paradigm. However, information processing is simply incorporated as an opportunistic by-product of routing in directed diffusion. While this might be sufficient for simple event monitoring applications where data volume is small, the lack of formal consideration of integrating information processing with routing makes the protocol inappropriate for applications with complex information processing. Some other geographic routing [65] and rumor routing [22] protocols also fall into this category.

The LEACH protocol [53] adopts a two-tier clustering structure, where the information processing is performed at each cluster head and routing is simply divided into two stages: routing from sensor nodes to cluster heads and from cluster heads to the base station. This is however, an empirical study that aims for energy-conservation by avoiding long distance communication but not really integrating information processing with routing.

A formal and analytical study of the impact of data aggregation on routing in WSNs is first presented by Krishnamachari *et al.* [68]. An intuitive theoretical bound is that if every k pieces of information can be aggregated into a single piece of information, the routing load can be

reduced by a factor of at most k. Here, the value of k is usually referred to as the *aggregation factor* or *correlation factor* among data.

Along this line, the impact of k on two different routing schemes, the Shortest Path Tree (SPT) and the Minimal Steiner Tree (MST) is investigated in [87]. It is further revealed that the optimal tree structure for integrated information processing and routing is a hybrid of SPT and MST. This can be explained by the intuition that SPT is optimal when k is one since the routing of each piece of source information becomes independent and MST is optimal when k is infinity since exactly one piece of information is transported on each edge of the tree.

The above conclusion on a hybrid routing scheme is also conformed by other results. Cristescu *et al.* assumes a simplified compression model [32], where the aggregation factor of a piece of information does not depend on the amount of side information, but only on its availability. A hybrid scheme of SPT and Minimal Spanning Tree, the Shallow Light Tree (STL) is proved to provide 2-approximation performance for minimizing the overall cost of the data gathering tree. For a very similar problem, when the joint entropy being modeled as a concave, but unknown function of the number of source nodes, a recursively clustering approximation algorithm is given by Goel *et al.* [47]. It is also noticed by the authors of [47] that the data gathering problem is essentially a single-source buy-at-bulk problem [9]. The key point is that the cost spent on each edge is a concave function of the number of source nodes that use this edge to communicate to the sink. Another randomized algorithm for routing information on a grid of sensors is proposed by [39] that achieves constant factor approximation (in expectation).

Moreover, some research efforts have investigated other routing substrates instead of a tree structure. Bo *et al.* proposed a distributed in-network processing algorithm that achieves maximal throughput under the assumption that the information processing is independent for all sources [56]. His algorithm is essentially based on the optimization of a network flow

problem. A hierarchical data gathering scheme for a linear array of sensor nodes is studied by ElBatt [36].

All the above papers [68, 47, 87, 32, 39, 56, 36] assume that certain coding mechanisms are available to accomplish the data aggregation operation, hence the authors can focus on a relatively high level algorithm design or performance analysis. Nevertheless, there are also papers that try to understand the inter-relationship between information processing and routing from information coding perspective.

Scaglione *et al.* considered tight coupling between routing and source coding for the problem of broadcast communication in a WSN subject to a given distortion constraint [97]. It is proved that while the whole network traffic for such a broadcast scales as $O(N \log N)$ (N being the number of sensor nodes), a simple integrated routing and source coding scheme can be used to reduce the traffic to $O(\sqrt{N})$, and hence is supportable by the network capacity which is also $O(\sqrt{N})$. It is also argued by Cristescu [32] that when Slepian-Wolf coding can be used without explicit communication, the shortest path tree is the optimal in terms of minimizing network traffic. However, to ease the task of a global Slepian-Wolf coding, the authors also proposed an approximation algorithm by grouping nodes into clusters and performing Slepian-Wolf coding in each cluster. In these works, the optimal clustering is conjectured to be NP-Hard.

Since the above works explicitly consider the collaboration between joint data compression and routing, the joint entropy of gathered information becomes a key factor that determines the problem settings and the consequent solutions. Hence, we refer to the routing techniques discussed above as entropy-aware routing.

There are also some other works also have been proposed for energy-efficient routing from a more general perspective. For example, the energy $\times$ delay metric is used to determine a data gathering substrate [73]; the expected number of transmissions (ETX) is used as a path metric for multi-hop transmission [34]; and the packet reception rate $\times$ distance metric is used

to choose a forwarding node during routing [101]. In this context, various tradeoffs including energy *vs* transmission delay, number of hops, path length have been explored.

1.5.5 Application Layer

At the application layer, both energy-efficient signal processing algorithms [5, 13] and adaptive-fidelity algorithms [110] have been studied for cross-layer optimization. The key idea is to trade the application-level information precision or accuracy for energy.

Another useful technique at application layer is the so-called tunable compression which tunes the compression ratio for balanced computation energy cost against communication energy cost. Consider the example of *gzip*. It provides up to ten levels of different compression ratio, with larger compression ratio resulting in longer compression time and hence higher energy cost [12, 17]. The use of tunable compression is focused on computation-intensive applications where traditional maximal compression becomes less energy efficient because of the over-paid computation energy for data compression. Hence, carefully choosing the compression ratio is necessary to explore the tradeoffs between computation and communication energy.

1.5.6 Putting It All Together

In Table 1.2, we illustrate the tradeoffs being explored by the aforementioned techniques. Note that it is often difficult, if necessary, to clearly isolate different performance metrics involved in the tradeoffs. For example, transmission delay and reliability are closely related at both physical layer and routing layer, since transmission delay depends on both the time cost for each transmission and the expected number of re-transmissions, which is determined by reliability. Also, the radio sleep scheduling at MAC layer affects both transmission delay and throughput simultaneously. Hence, in many cases, it is necessary and helpful to understand the tradeoffs while taking multiple performance metrics into consideration.

Table 1.2: Examples of cross-layer optimization techniques and the associated tradeoffs

System layer	Techniques	Tradeoffs
Application	Energy-efficient signal processing	Energy *vs* information precision/accuracy
	Adaptive fidelity algorithms	Energy *vs* information precision/accuracy
	Joint routing and coding	Traffic *vs* delay
	Tunable compression	Energy *vs* output size
Routing	Energy-aware routing	Energy *vs* delay/reliability/path length
	Entropy-aware routing	Energy *vs* delay/routing complexity
MAC	Radio sleep scheduling	Energy *vs* delay/throughput/topology
Physical	Power control	Energy *vs* connectivity/topology/reliability
	Rate adaptation	Energy *vs* delay
	Adaptive coding	Energy *vs* delay/reliability
Hardware	Low-power circuit	Energy *vs* performance
	Node sleep scheduling	Energy *vs* response time/sensing coverage
	Voltage scaling	Energy *vs* delay

Moreover, the concrete interpretation of a single performance metric may vary across different levels. For example, delay at application layer often refers to the end-to-end time duration for performing a specific task, e.g., gathering information across the network. At routing layer, delay usually refers to the time duration of transporting a packet over a path between two sensor nodes. At physical layer, delay may refer to the time duration for packet transmission over a single link. However, link-wide delay at physical layer and path-wide delay at routing layer also affect system-wide delay at application layer. Due to such an inherent relationship, we do not rigorously distinguish between these different interpretations.

Based on the table, we summarize two important issues in cross-layer optimization, which also reinforce our motivation stated in Section 1.4.1.

First, it is worth noting that these optimization techniques are not independent. In fact, the behavior of certain techniques can change the optimization space and hence solution for other techniques. For example, the radio sleep scheduling at MAC layer affects network topology, which in turn impacts the routing decision at routing layer. Also, the sleep scheduling affects channel access at MAC layer and hence signal interference at physical layer, which is referenced

while applying power control and rate adaptation techniques. Given such a complex inter-relationship, it is important to understand the impact of certain techniques across various stacks before applying it.

Second, one single performance metric observed by the users can be affected by techniques across different layers. For example, network topology is affected by both physical layer power control and MAC layer sleep scheduling. Also, application-level delay is co-determined by a series of techniques, including application layer joint routing and coding scheduling, routing layer decision, MAC layer sleep scheduling, physical layer packet transmission, and hardware layer CPU processing. If an application-level delay constraint is imposed by the user, the up front question to explore the energy *vs* delay tradeoff at these different layers is how to break the application-level delay constraint into sub-constraints for each individual layer. There is no easy answer for this question unless a cross-layer optimization technique can be developed to integrate the tradeoffs at different layers.

1.6 Research Contributions of this Thesis

The main theme of this thesis is on algorithm development and performance analysis for cross-layer optimization for energy-efficient information processing and routing in WSNs.

While our research efforts are stemming from the general concept of information processing and routing, this thesis covers the following three specific topics:

1. information processing within a cluster of sensor nodes (or in-cluster processing),
2. information transportation over a given multi-hop tree structure, and
3. information routing for computation-intensive applications over a general graph.

Besides that each of these three topics is important and challenging in itself, the composition of them gives a complete operating flow of information processing and routing. This is the major motivation to choose them as the research topics in this thesis.

In our research efforts towards the above topics, we are particularly interested in three system knobs, including voltage scaling, rate adaptation, and tunable compression. While voltage scaling and rate adaption trades the computation/communication speed against computation/communication energy, tunable compression is a bridge that enables the tradeoffs between computation and communication. We illustrate the tradeoffs that are explored by these three system knobs in Figure 1.2. We will give detailed energy models for these three system knobs in Chapter 2.

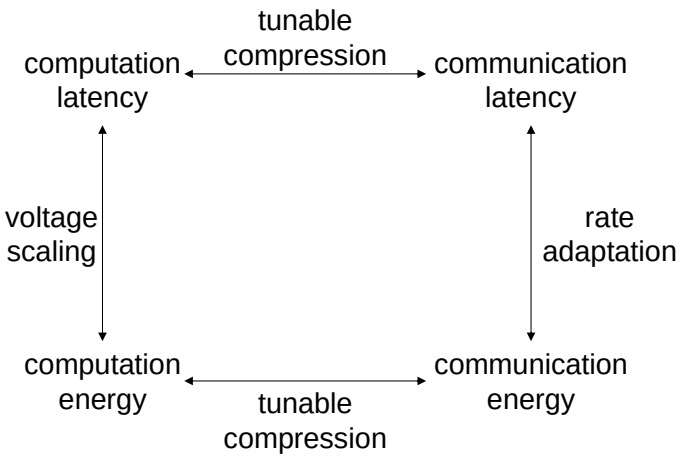


Figure 1.2: Tradeoffs explored by three system knobs: voltage scaling, rate adaptation, and tunable compression

These three system knobs are applied in the aforementioned research topics. In the first research topic, we investigate the application of voltage scaling and rate adaptation to maximize the system lifetime for in-cluster processing. In the second topic, we study rate adaptation for

minimizing the energy cost for information transporting over an existing tree. In the last topic, we show that tunable compression can be incorporated with the construction of a routing tree for minimizing the overall computation and communication energy in information routing.

One example of applying our research efforts is the cluster-based network scheme [30, 52, 106, 130, 129]. In this scheme, the whole network is partitioned into either static and dynamic clusters, probably with one sensor node within each cluster designated as a cluster head. We assume that each cluster behaves as a basic function unit where in-cluster processing is responsible for converting raw data into useful information. The processed information is then transported back to the base station through either direct communication from cluster heads to base station [52], or a multi-hop tree that consists of only cluster heads [106], or a general multi-hop tree that may consist of any types of sensor nodes in the network [87, 130]. While the construction of such an cluster-based infrastructure is beyond the scope of this thesis, we can see that our three research topics fit well into this scheme. Moreover, the proposed techniques are applicable for other scenarios as well.

Note that our research efforts by no means provide a complete solution for information processing and routing. Our work is based on a relatively high model of the system. We are not concerned about details of specific hardware to realize the system knobs, or protocols for MAC layer scheduling and networking layer communication, or techniques for signal processing and data compression. Our focus is to improve the energy-efficiency of the systems by assuming that all such techniques are available. From the system stack perspective, based on where the low level system knobs are located, our work sits between the hardware/MAC/routing layer and application layer when voltage scaling, rate adaptation, or tunable compression are applied, respectively.

Before proceeding to detailed description of our research efforts, we first describe the two classes of targeting applications.

1.6.1 Two Classes of Target Applications

Our research is motivated by the following two classes of applications.

1.6.1.1 Real-time Information Gathering Applications

The first class of applications focuses on real-time environment monitoring. The very basic functionality is to periodically gather certain information from the environment, such as temperature and humidity. We note that many of such applications require timely delivery of the gathered data. For example, a seismic detection system or a forest fire alarm system may require the gathered data on vibration and temperature to be processed and sent to the end user within 1 second after the data is sensed. Hence, it is necessary to incorporate certain real-time requirements into the design of such mission-critical applications. Specifically, we consider an end-to-end latency constraint, which is imposed on the delay for processing and transporting the data.

To model such a latency constraint in the above cluster-based infrastructure, we partition the latency constraint into two pieces — one for the information processing within each cluster, and the other for the transportation of information from clusters to the base station. For the sake of our study, we assume that such a partition is given. Hence, we can focus on using the voltage scaling and rate adaptation technique to explore the tradeoffs between latency and energy cost during the course of in-cluster processing and the information transportation, respectively. The key point is to deliver "just enough" performance in terms of latency so that the energy cost can be minimized.

The above real-time scenario has been studied in several papers [77, 18, 21]. The epoch-based data [77] gathering is one example in which one round of data gathering should be performed within each epoch. In other words, the length of the epoch equals the end-to-end latency

constraint. Also, the realization of real-time constraint requires the use of time-synchronization techniques, which have been studied by [38, 44, 113].

1.6.1.2 Computation-Intensive Data Streaming Applications

The second class of applications emphasizes on the processing and transportation of large volume of data, such as the monitoring of complicated manufacturing systems, video surveillance, and video streaming. The processing of large volume of data is also possible when sensed data are accumulated for a long time before it is transported to the base station. In these cases, the computation cost becomes comparable to communication cost. Hence, the traditional maximum data compression for saving communication energy at the cost of computation energy might be a bad choice due to overpaid computation energy. A balance between computation energy and communication is thus desired.

One example of the above class of applications is the video monitoring of volcanos. Imagine a set of camera-equipped sensor nodes that are placed around a volcano. The images taken by the cameras are then needed to be transported to the base station. If the density of the camera-equipped sensor nodes are sufficiently high, the images taken by the cameras will have large correlation which effects joint data compression to remove the redundancy among the images taken by multiple cameras. However, the processing of images often results high computation energy cost. Hence, by exploiting the tradeoffs between computation energy and communication energy using tunable compression, we can find an appropriate routing scheme with joint compression strategy for minimizing the total energy cost.

1.6.2 In-Cluster Information Processing

We consider the scheduling of an application graph for information processing in a collocated cluster [46], where all sensor nodes are connected through one-hop communication. This includes the *assignment* of tasks onto sensor nodes, the *scheduling* of computation and communication tasks, and the *setting* of computation or communication speed of each task. We consider a real-time scenario where a latency constraint is imposed for the whole process. The problem is NP-hard in general.

In this problem, we are mainly dealing with in-cluster information processing, while the routing decision is not concerned. Our contributions include:

1. A novel objective function of balancing the energy cost of all sensor nodes within the cluster to avoid "hot spot" of the cluster and hence maximize the lifetime of the cluster.

2. The formulation of the problem using Integer Linear Programming (ILP) that provides useful performance benchmark for small scale problems.

3. A 3-phase heuristic that achieves considerable energy conservation — up to 10x lifetime improvement for synthesized task graphs, and up to 8x lifetime improvement for the LU factorization algorithm and up to 9x improvement for Fast Fourier Transformation (FFT).

1.6.3 Information Transportation over a Given Tree

We incorporate rate adaptation into information transportation over a given routing substrate, e.g., a data gathering tree, subject to a latency constraint. We assume that the data correlation model among multiple source nodes is given so that the flow on each edge in the tree is determined by certain data aggregation mechanism a priori.

In this problem, we assume the time and energy cost for data aggregation is negligible and hence we focus on the adjustment of the transmission rate over the tree substrate so that the total energy cost is minimized. Our contributions include:

1. First work to apply rate adaptation on a tree-structure network substrate.
2. Off-line optimization algorithms based on iterative numerical optimization and dynamic programming.
3. A greedy policy based on-line protocol that relies on local information only.
4. Up to 90% energy savings achievable by using the proposed techniques in simulations, compared to the baseline where all transmissions are performed using the highest transmission rate.

1.6.4 Information Routing with Tunable Compression

Our third research effort considers the problem of constructing a tree for information processing and routing of computation-intensive data streaming applications. We use tunable compression to avoid over-compression of data so that a balanced computation energy *vs* communication energy can be achieved.

In this problem, information processing is captured by the adjustment of compression ratio while the information routing is determined by the constructed tree. Due to the coupling of these two factors, the problem is NP-hard in general. Our contributions include:

1. First work to formally consider computation cost for information processing and routing.
2. A data streaming model that facilitates the application of tunable compression.
3. Optimal flow over a given path with lower bounds of flow specified for each edge on the path.

4. Performance study for shortest path tree (SPT) and the minimal steiner tree (MST) in grid deployment that indicates the important tradeoffs between these two tree structures.

5. An approximation algorithm for general graph settings based on metric approximation.

1.6.5 Summarize

Our research efforts are based on a relatively high level system models such that details about low-level hardware knobs, MAC scheduling, networking protocols, and signal processing techniques are effectively abstracted away. Specifically, our effort on in-cluster information processing (Section 1.6.2) assumes that an application graph for information processing within a single-hop cluster is given. Such an application graph can be used to capture various distributed signal/image processing algorithms. Our second research effort (Section 1.6.3) assumes that a routing tree has been set-up by, for example, any of the techniques mentioned in Section 1.5.4 and that the joint data compression techniques on all sensor nodes are given as well. We further assume that the aggregation factor at all nodes are known a priori, implying that we know the flow size on all edges in the tree. Our third effort (Section 1.6.4) relaxed the assumptions in the second research effort about a given tree and compression technique. Instead, we assume that (1) the routing tree needs to be structured from an arbitrary graph and (2) the data compression techniques are tunable in terms of compression time and ratio. In this context, we try to understand the inter-relationship between tunable compression and routing decision.

Regarding cross-layer optimization, we have investigated three important system knobs in our research. Our first research effort studies the application of voltage scaling and rate adaptation in a distributed wireless system. We address several novel challenges originated from WSNs such as task placement constraint and packet scheduling for collision-free communication. Our second research effort demonstrates the application of rate adaptation over a routing tree, where the challenge is to corporate communication dependency into packet scheduling. Both of

these two research efforts focus on exploring the tradeoffs between energy cost and application-level latency. Finally, our third research effort shows the integration of tunable compression with the construction of a routing tree. This is a novel research topic that targets the tradeoff between computation energy and communication energy.

1.7 Thesis Organization

The rest of the thesis is organized as follows.

In Chapter 2, we we give a list of common notations that are used throughout this thesis and then describe the energy models for three system knobs: voltage scaling, rate adaptation, and tunable compression. We will also demonstrate the tradeoffs that are involved in these system knobs. In Chapters 3 to 5, we describe our three research efforts in detail. In Chapter 6, we give concluding remarks and a summary of future directions.

Chapter 2

Energy Models

In this chapter, we describe the energy models for voltage scaling, rate adaptation, and tunable compression.

2.1 Definitions and Notations

Before we describe the energy models, we first give a list of basic definitions and notations that will be used throughout the thesis.

2.1.1 Mathematics and Graphs

Let $\mathbb{N}^+$ denote the set of positive natural numbers, i.e., $\mathbb{N}^+ = \{1, 2, \dots\}$. We usually use i, j, k to denote the indices of a set or an array. Given $i \in \mathbb{N}^+$, let $[i]$ denote the set $\{1, 2, \dots, i\} \subset \mathbb{N}^+$.

Let $G =< V, E >$ denote a graph with vertex set V and edge set E. If G is undirected, we use (V_i, V_j), or simply (i, j) to denote an edge that connects vertices V_i and V_j. If G is directed, let (V_i, V_j), or simply (i, j) denote an edge that points from V_i to V_j. In this thesis, we will often use $u \in V$ to denote a node, and $e \in E$ to denote an edge, if the indices of nodes or edges can be ignored from the context.

In the following, we define the predecessor and successor relationship for nodes and edges in a directed graph.

Definition 1 *Given two nodes V_1 and V_2 in a directed graph $G =< V, L >$, V_1 is a predecessor node (or simply predecessor) of V_2 if there is a path from V_1 to V_2, denoted as $V_1 \prec V_2$. We also refer to V_2 as a successor node (or simply successor) of V_1.*

Definition 2 *Given two edges L_1 and L_2 in a directed graph $G =< V, L >$, L_1 is a predecessor edge (or simply predecessor) of L_2 if the ending point of L_1 is a predecessor of the starting point of L_2, denoted as $L_1 \prec L_2$. We also refer to L_2 as a successor edge (or simply successor) of L_1.*

Since general network topology is usually modeled as an undirected graph, the above predecessor/successor relationship is usually applied in a specific routing substrate such as a data gathering tree represented as a directed graph. For such a data gathering tree, the predecessor/successor relationship defines the order of edges that a packet may traverse within the tree towards the root. We will also use directed graph to model the inner-structure of an application. In this case, the predecessor/successor relationship specifies the order of execution for the involved tasks in the application. We will explain this in Section 2.1.3.

2.1.2 Network Topology Graph

Consider a network of sensor nodes. Given $v \in \mathbb{N}^+$, we use V to denote the set of v sensor nodes, i.e., $V = \{V_i : i \in [v]\}$. Given $l \in \mathbb{N}^+$, we use L to denote the set of l communication links, i.e., $L = \{L_i : i \in [l]\}$.

Definition 3 *An network topology graph (or simply network graph) $NG =< V, L >$ is a graph with vertex set V representing the set of v sensor nodes and edge set L representing the set of l communication links.*

NG is undirected if the communication links are symmetric and directed if the communication links are asymmetric. Throughout this thesis, we consider symmetric communication links only. We use $(i, j) \in L$ to denote a communication link that connects sensor nodes V_i and V_j. For the ease of presentation, the terms "sensor node", "node", and "vertex" are often used exchangeably. Also, the terms "communication link", "link", and "edge" are often used exchangeably. In the following, we define several special network graphs.

Definition 4 *A collocated network of sensor nodes consists of a set of sensor nodes that are connected to each other via one-hop communication.*

In other words, the network graph for a collocated network is a complete graph.

Definition 5 *A grid network of sensor nodes consists of a set of sensor nodes that are placed on a grid structure, where each node can communicate to its eight neighbors (ignoring boundary effects) through one-hop communication.*

Definition 6 *An arbitrary network of sensor nodes consists of a set of sensor nodes that are randomly placed on a planar. Each node can communicate to another node within distance d through one-hop communication, where d is given as the communication radius.*

To model the energy cost for communication between any pair of sensor nodes in V, we also associate a weight with each link in L, which reflects the energy cost for transmitting a packet of unit size over the link. In this thesis, we will have different assumptions about the weights according to the specific problem scenarios.

For example, in Chapters 3 and 4, we will model the weight as a function of the transmission speed and radius, and the communication environment. In Chapter 5, we will assume that the transmission speed is fixed so that the weight simply becomes a scalar value for each edge, which is determined by the transmission radius and communication environment. Moreover, in the special case of grid network in Chapter 5, we assume that the cost for transmitting a

packet of unit size over any communication link (either horizontal, vertical, or diagonal) is the same.

Since the above different network topologies are used in Chapters 3, 4, and 5, we will give detailed description of the link weights in these chapters accordingly.

2.1.3 Application Graph

Given $c \in \mathbb{N}^+$, we use C to denote a set of c computation tasks, i.e., $C = \{C_i : i \in [c]\}$. Given $q \in \mathbb{N}^+$, we use Q to denote a set of q communication tasks, i.e., $Q = \{Q_i : i \in [q]\}$. Since a WSN application usually consists of a set of computation tasks and a set of communication tasks, we often use a graph representation to abstract the dependency and inter-relationship between the computation and communication tasks. Such a graph is referred to as an *application graph*, which is defined as follow:

Definition 7 *An application graph $AG =< C, Q >$ is a directed acyclic graph (DAG) with vertex set C representing the set of c computation tasks and edge set Q representing the set of q communication tasks.*

We often use tasks to refer to both computation and communication tasks. Also, for a communication task, the terms "sending node" and "sender" are used interchangeably. So are the terms "receiving node" and "receiver".

In the following, we slightly abuse the predecessor/successor relationship to define the dependency relationship between a pair of computation/communication tasks or between a computation task and a communication task.

Definition 8 *Any edge $Q_k = (C_i, C_j)$ in an application graph defines a dependency relationship between tasks C_i, Q_k, and C_j, which specifies that (1) the execution of Q_k cannot start until C_i is finished, denoted as $C_i \prec Q_k$ and (2) the execution of C_j cannot start until Q_k is finished, denoted as $Q_k \prec C_j$.*

Since the dependency relationship is essentially a predecessor/successor relationship, the dependency relationship is also transitive, i.e., $C_i \prec Q_k$ and $Q_k \prec C_j$ imply $C_i \prec C_j$. We refer to C_i as a *predecessor* of C_j and C_j, a *successor* of C_i. Similarly, consider a task C_k with incoming edge Q_i and outgoing edge Q_j, we have $Q_i \prec C_k$ and $C_k \prec Q_j$, which imply $Q_i \prec Q_j$. Extending the above data dependency, the execution of a task can only be started after it receives output from all of its predecessors, if any.

Definition 9 *A computation task without any predecessors is referred to as a* source *task. A computation task without any successors is referred to as a* sink *task.*

The above definition of dependency is mainly a data dependency — the output of C_i acts as an input of C_j. If C_i and C_j are hosted on two different sensor nodes, then it is necessary to transmit the output of C_i to the sensor node where C_j is hosted before the execution of C_j. If C_i and C_j are hosted on the same sensor nodes, data exchange is performed through memory exchange, which has a negligible cost compared with packet transmission.

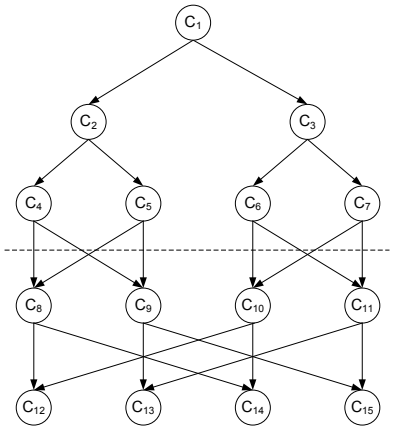
Note that we do not model control dependency with the above application graph. This is reasonable if we abstract computation tasks at a reasonably coarse granularity and consider the worst case execution time only. To model control dependency cause the application model to be unnecessarily complex from a high level abstraction. In fact, application graph with data dependency only is widely used in parallel and distributed computing [126, 48, 23] and real-time computing [63, 42, 57, 76].

Similar to the weights in network graph, we also annotate the computation and communication loads of tasks by giving weights to tasks in the application graph. For a computation task, the associated weight is the worst case number of CPU cycles that are needed to execute the task. For a communication task, the associated weight is the size of the packet needed to be transmitted. We will give more detailed description and notations of task weights in Chapters 3, 4, and 5 respectively based on specific problem scenarios.

The above application graph can be used to represent both signal processing algorithm and information processing and routing procedure. For example, the recursive, one-dimensional Fast Fourier Transformation (FFT) algorithm is given in Figure 2.1(a) with an example task graph of 4 points demonstrated in Figure 2.1(b).

FFT(A, ω)
1. Set $l = \text{length}(A)$
2. **If** $l = 1$, return A
3. $Y^{(0)} = \text{FFT}((A[0], A[2], \ldots, A[l-2]), \omega^2)$
4. $Y^{(1)} = \text{FFT}((A[1], A[3], \ldots, A[l-1]), \omega^2)$
5. **For** $i = 0$ to $l/2 - 1$ **Do**
6. $\quad Y[i] = Y^{(0)}[i] + \omega^i \times Y^{(1)}[i]$
7. $\quad Y[i + l/2] = Y^{(0)}[i] - \omega^i \times Y^{(1)}[i]$
8. **Return** Y

(a) sequential algorithm



(b) example application graph with 4 points

Figure 2.1: Fast Fourier Transformation (FFT) algorithm

Another example is the information transportation over a given routing tree substrate. Intuitively, the structure of the application graph is identical to the structure of the routing tree. For the illustrating task graph shown in Figure 2.2, each task executes a data aggregation operation. Note that in general, an application graph for information transportation does not need to have a tree structure.

The task graph for the operation of information routing without a given routing structure is however tricky. Basically, the structure of the task graph cannot be pre-determined until the structure of the routing tree is determined. In this case, the aforementioned task graph is insufficient to describe the application.

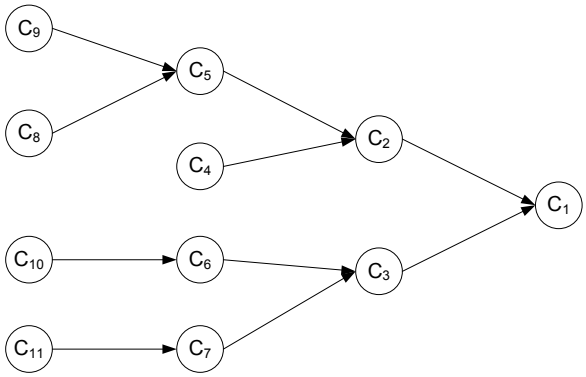


Figure 2.2: Application graph for information transportation over a given tree

2.1.4 Performance Measurement

We are interested in several performance measurements. First of all, energy is a basic concern. We usually use ϵ_i to denote the energy cost of either a computation task or communication task, where i is the corresponding index of the computation or communication task in the application graph. Since we will consider the scenario for real-time processing, we use τ_i to denote the time cost of a computation or communication task i. This time cost is sometimes referred to as delay, or latency. Also, the subscription i is often ignored if it is not important in the context.

In real-time scenarios, the accumulated time cost of tasks is often limited by certain constraint at application level. We refer to such a constraint as the *latency constraint*, denoted as Γ. Such a latency constraint is common in literature of real-time task scheduling [63, 42, 57, 76].

The notion of latency constraint is usually used together with the notion of periodicity in real-time task scheduling. An application (may consists of exactly one task) is periodic if it needs to be repeatedly executed every k time units, where k is defined as the period of the application. On the other hand, an application is transient if it is executed only once. The

latency constraint can be applied for both periodic or transient applications. For transient application, the latency constraint is easy to understand. For period applications, the latency constraint is considered to be less than the application period in our case. When the latency constraint is larger than the period, certain pipeline techniques need to be developed so that the execution of two or more rounds of information processing and routing is overlapped with each other, which is beyond the scope of our research.

A similar concept with periodic application in WSNs is the epoch-based system [77]. Within each epoch, the desired system functionality (e.g., a query) need to be performed. From this perspective, the length of the epoch can be understood as the latency constraint. Moreover, to measure the system delay, we assume the availability of time-synchronization schemes within the network (e.g., [38, 44, 113]).

Some other performance measurements, such as throughput, have also been studied in literature [56]. Our research are focused on applications that require low-duty cycle, for which throughput optimization is not a concern.

2.2 Energy Models

In this section, we will describe the energy models for the three system knobs under consideration. For voltage scaling, the model is developed at the circuit switch level. For rate adaptation, the model is developed at the physical level. For tunable compression, the model is developed at the application level.

Please note that from the perspective of optimization at a relatively high abstract level, we are more interested in the high-level tradeoffs enabled by the system knobs rather than the details of low level implementation that will be presented in this section. Such tradeoffs are often abstracted as a convex function. For example, the energy cost is usually a convex function

of either the computation time or the communication time. However, for practical reason, we shall still carefully understand the details of these knobs in this section.

2.2.1 Voltage Scaling

There are three major components of power consumption for executing a task by a CMOS integrated circuit: switching power, short-circuit power, and leakage power [25]. We focus on the switching power that dominates the power consumption for state-of-the-art techniques.

Let V_{dd} denote the supply voltage, f_{clock} denote the clock frequency, C_L denote the loading capacitance, and P_t denote the probability that a power-consuming transition occurs (the activity factor). The switching power P_{sw} can be modeled as:

$$P_{sw} = p_t \times C_L \times V_{dd}^2 \times f_{clock} \ . \tag{2.1}$$

The product of $p_t \times C_L$ is also referred to as the effective switched capacitance.

From (2.1), we observe that the power consumption increases quadratically with supply voltage. Hence, the power consumption can be reduced by decreasing the supply voltage. However, this comes at the expense of reduced processing speed, and hence, increased processing time τ which is given by:

$$\tau = k_c \frac{V_{dd}}{(V_{dd} - V_T)^2} \ , \tag{2.2}$$

where k_c is a constant and V_T is the threshold voltage [25].

Based on the above models, we can see that $P_{sw} \propto V_{dd}^2$ and $\tau \propto \frac{1}{V_{dd}}$. Hence, the energy cost $\epsilon = P_{sw} \times \tau$ increases linearly with V_{dd}. In other words, the energy cost for executing a task can be reduced at the expense of increased execution delay. In fact, the energy cost can be modeled as a monotonically decreasing and strictly convex function of the delay.

Since delay is also understood as the reciprocal of the processing speed, we can also model the energy cost of processing a task as an increasing function of processing speed [24], which is detailed in Chapter 3.

Note that switching voltage is not free — it takes both time and energy. While most of the techniques proposed for voltage scaling ignore this fact or assume that such time and energy cost are absorbed in the execution cost of tasks, there are few papers that explicitly address this cost [57].

2.2.2 Rate Adaptation

We model the transmission energy using the example of modulation scaling [98] based on the Quadrature Ampitude Modulation (QAM) scheme [117]. Consider a communication task that transmits a packet of s bits between two sensor nodes. Assuming that the symbol rate, R_s, is fixed, the transmission time, τ, can be calculated as [98]:

$$\tau = \frac{s}{b \cdot R_s} , \tag{2.3}$$

where b is the modulation level of the sender in terms of the constellation size (number of bits per symbol).

The corresponding transmission energy can be modeled as the sum of output energy and electronics energy, which is also determined by b. To illustrate the key energy-latency tradeoffs, we abstract the energy cost as a function of τ [98], denoted as $w(\tau)$:

$$w(\tau) = [C_{tr} \cdot (2^{\frac{s}{\tau \cdot R_s}} - 1) + C_{ele}] \cdot \tau \cdot R_s , \tag{2.4}$$

where C_{tr} is determined by the quality of transmission (in terms of Bit Error Rate) and the noise power, and C_{ele} is a device-dependent parameter that determines the power consumption

of the electronic circuitry of the sender. Further, the output power, P_o, and the electronic power, P_e, can be modeled as follows [98]:

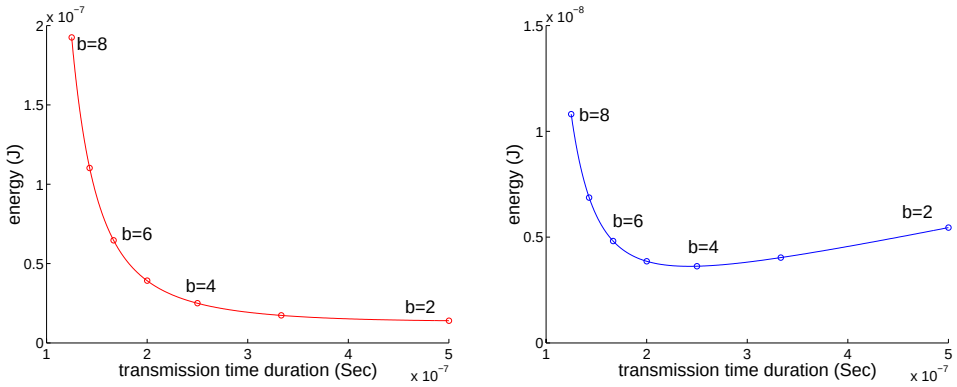
$$P_o = C_{tr} \cdot R_s \cdot (2^b - 1) \text{ and} \quad (2.5)$$

$$P_e = C_{ele} \cdot R_s \ . \quad (2.6)$$

Note that different assumptions about the radio characteristics, including power consumption and data rate, may significantly affect the analysis of various energy-saving mechanisms. In this work, we consider the radio modules described in [93, 120]. Typically, for *short-range* communication with $R_s = 1$ Mbaud, the electronic power of the radio is approximately 10 mW, while the output power is approximately 1 mW at 4-QAM (which is translated into 2 Mbps). Note that the above data rate and power consumption are better than currently available radios for commercial sensor nodes that typically support data rate up to 100 Kbps with slightly higher power characteristics, such as Berkeley motes. However, radio devices with the above specifications are anticipated in the near future.

From the calculation of P_o and P_e, it can be derived that $C_{tr} \approx 3 \times 10^{-10}$ and $C_{ele} = 10^{-8}$. Further, we consider a d^2 path loss model for signal propagation, where d is the communication radius. Assuming that it takes 10 pJ/bit/m^2 by the amplifier to transmit one bit at an acceptable quality [52], we infer that the corresponding communication radius for 1 mW output power is $\sqrt{50} \approx 7$ m (from $\frac{1 \text{ mW}}{2 \times 10^6 \text{ bit/sec}} = 10 \text{ pJ/bit/m}^2 \times d^2$). In our study, we also consider one more case of communication in WSNs with longer radius. Specifically, we set the communication range to 30 m, implying an output power of 10 pJ/bit/m$^2 \times 30^2$ m$^2 \times 2 \times 10^6$ bit/sec = 18 mW at 4-QAM, and consequently $C_{tr} = 6 \times 10^{-9}$. We refer to this communication scenario as *long-range* communication. Note that these numbers for communication radii are for illustrative purpose only – to show the different weights of C_{tr} against C_{ele} with respect to variations

in communication radius. They may vary according to different radio devices and operating environments.



(a) Long-range communication

(b) Short-range communication

Figure 2.3: Energy-latency tradeoffs for transmitting one bit

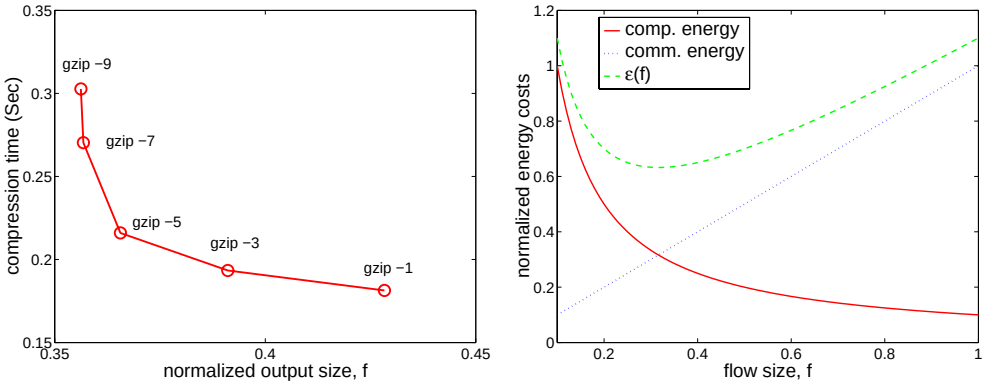
Figure 2.3 plots the energy functions with $b \in [2, 8]$ for the long and short range communication scenarios. In practice, b is typically set to positive even integers, as indicated by circles in the figure. We can observe a 10-time energy reduction for long-range communication by varying b from 8 to 2 and a 3-time energy reduction for short-range communication by varying b from 8 to 4. Intuitively, it is more beneficial to explore the energy-latency tradeoffs for the long-range communication.

Though QAM is used as an example for abstracting the energy model, the algorithms presented in our research are extendible to other modulation schemes and techniques that can be used to trade latency against energy, such as code scaling [7].

2.2.3 Tunable Compression

The concept of *tunable compression* is not new. For example, the well-known *gzip* program for lossless data compression supports up to ten levels of different compression ratio, with larger compression ratio resulting in longer compression time and hence higher energy cost [12, 17].

Since it is quite difficult to define a general form for characterizing the energy costs of various compressing schemes, we use a simple model that captures the principle rationale, which states that the computation time complexity of compressing one unit data is inversely proportional to the output size. Further, the energy cost is proportional to the time complexity [12]. We illustrate the above rationale using the example of *gzip* to compress the benchmark file "alice29.txt" from the Canterbury Corpus [17] at 5 different levels of compression ratio (by properly parameterizing *gzip*). The curve of running time *vs* the normalized output size over input size is shown in Figure 2.4(a) (averaged over 20 runs on a SUN SPARC II machine). In fact, similar compression time *vs* normalized output size tradeoffs are observed for a collection of various compression techniques [12].



(a) Compression ratio *vs* compression time for *gzip*

(b) Computation energy *vs* communication energy tradeoffs on a single link ($w_e = 1$, $\gamma = 0.1$)

Figure 2.4: Energy tradeoffs by tunable compression

We define a pre-specified system parameter, $\gamma \geq 0$ that abstracts the relative energy cost of compressing one unit data normalized by the cost of communicating one unit data. Following

the above rationale and for the purpose of illustration, the energy cost of compressing a data packet of size s to an output of size f is modeled as function

$$g(f) = s\gamma(\frac{s}{f}) \ . \tag{2.7}$$

While the term s indicates that the energy cost is proportional to input size, the term $\frac{s}{f}$ signifies that the energy cost is also inversely proportional to the compression ratio. We will consider input of unit size in this thesis. Hence, (2.7) can be simplified to $g(f) = \frac{\gamma}{f}$.

We now illustrate the tradeoffs between computation and communication energy using the example of a one-hop link. Let $e = (u, v)$ denote the link, where u generates one unit data that needs to be transmitted to v after appropriate compression by u. Let f denote the output data size, i.e., the flow over e, which is lower bounded by the entropy of the unit of data, denoted as ρ. Let w_e denote the cost of transmitting a unit data packet over e. The overall energy costs, denoted as $\epsilon(f)$ can be modeled as a function of f:

$$\epsilon(f) = \frac{\gamma}{f} + f \cdot w_e \ . \tag{2.8}$$

We plot $\epsilon(f)$ in Figure 2.4(b) with $w_e = 1$, $\gamma = 0.1$ and varying f from 0.1 to 1 (we omit the boundary effect of ρ as for now). Intuitively, $w_e = 1$ means that to transmit one unit data costs one unit energy. Since the energy of transmitting one bit is typically around the energy of executing 1000 instructions for contemporary hardwares [93], the realistic meaning behind $\gamma = 0.1$ is that around 100 instructions need to be executed for generating each bit in the output.

Clearly, $\epsilon(f)$ is convex and the minimum is achieved when $\epsilon'(f) = 0$, where $\epsilon'(f)$ is the first derivative of $\epsilon(f)$. Let f_0 denote the desired flow with $\epsilon'(f_0) = 0$. By simple algebra

manipulation, we have $f_0 = \sqrt{\frac{\gamma}{w}}$. Considering the boundary effects of ρ, the optimal value of f equals f_0 if $\epsilon'(\rho) \leq 0$ and $\epsilon'(1) \geq 0$, or ρ if $\epsilon'(\rho) \geq 0$, or 1 if $\epsilon'(1) \leq 0$.

Chapter 3

Information Processing within a Collocated Cluster

In this chapter, we investigate the application of voltage scaling and rate adaptation for information processing in a collocated cluster with latency constraint.

3.1 Overview

While information processing in wireless sensor networks is typically distributed throughout the network, a practical strategy is to organize sensor nodes that are geographically close to each other into small groups with each group behaving as a basic unit for collaborative processing of gathered information. For instance, in a target tracking application, up to thousands of sensor nodes are dispersed over a specific area of interest. The sensor nodes are usually organized into clusters [52, 130] with each cluster consisting of tens of sensor nodes. Distributed signal detection and collaborative data processing are performed within each cluster for detecting, identifying, and tracking vehicles. Some of the operations involved in such data processing include the LU factorization [28] and the Fast Fourier Transformation (FFT) [29].

In this chapter, we consider clusters that each consists of a collocated network of sensor nodes. We assume that all sensor nodes are equipped with both voltage scaling and rate adaptation. Also, multiple channels are available for communication within the cluster.

The information processing within the cluster is abstracted as a periodic application graph with delay constraint. We use four components to specify the resulting execution of the application, including (1) the *assignment* of computation tasks onto sensor nodes and communication tasks onto channels, (2) the *voltage settings* of computation tasks, (3) the *rate setting* of communication tasks, and (4) the *scheduling* of computation and communication tasks. We refer to the combination of instantiations of the above four components as a *task allocation*.

Besides the latency constraint, we consider two more constraints. The first one is the *exclusive access constraint*, which specifies that a non-preemptive scheduling policy is employed by each sensor node and each wireless channel. Also, at any time, a sensor node can receive or send data by using at most one channel. The underlying network protocol is assumed to be capable of scheduling a communication task over a specified channel according to the start and finish time of the task. Such a scheduling policy requires coarse-level bandwidth reservation mechanisms, which can be provided by, for example, a TDMA protocol. Moreover, we consider the *task placement constraint*, which is typically required when certain tasks for sensing the raw data must be allocated onto different sensor nodes.

Our general goal is to find a task allocation such that the lifetime of the cluster is maximized. To realize such a goal, we propose an energy-balanced execution scenario such that the maximal energy cost among all sensor nodes is minimized, subject to the above latency constraint, exclusive access, and task placement constraints.

3.1.1 Our Contributions

The idea of energy-balanced task allocation for information processing with a cluster of collocated network in WSNs is proposed. As we shall see in Section 3.2, most of the previous efforts in energy-aware task allocation or resource management try to minimize the overall energy cost of the system. This strategy may not be suitable in the context of WSNs, since each sensor node is equipped with its own energy source. Moreover, for event-driven systems, applications often need to be executed after the system has been working for sometime. In such a case, an energy-balanced task allocation should also consider the fact that the remaining energy can vary among sensor nodes.

To the best of the authors' knowledge, this is the first work for task allocation in WSNs that considers the time and energy costs of both the computation and communication tasks. We first present an integer linear programming (ILP) formulation of our problem. The optimal solution of the problem can be obtained by using a commercial software package such as LINDO [72], though the running time of such a software can be large. Hence, we propose a polynomial time 3-phase heuristic.

We first present simulation results that only consider voltage scaling. Our simulation results show that for small scale problems, up to 5x lifetime improvement is achieved by the ILP-based approach, compared with the case where no voltage scaling is used. Also, the 3-phase heuristic achieves up to 63% of the system lifetime obtained by the ILP-based approach. For large scale problems, the 3-phase heuristic achieves up to 3.5x lifetime improvement when only voltage scaling is used. By incorporating rate adaptation, up to 10x lifetime improvement was observed. Simulations were also conducted for application graphs from two real world problems – LU factorization and FFT. We observed a lifetime improvement of up to 8x for the LU factorization algorithm and up to 9x for FFT.

3.1.2 Chapter Organization

We discuss the related work in Section 3.2. The energy-balanced task allocation problem is defined in Section 3.3. The ILP formulation of the problem is given in Section 3.4. The 3-phase heuristic is described in Section 3.5. Simulation results are demonstrated in Section 3.6. Finally, we give concluding remarks in Section 3.7.

3.2 Related Work

Extensive research efforts have studied the problem of energy-efficient task allocation and scheduling with voltage scaling in uni-processor real-time systems, including [10, 57, 104, 127]. Recently, research interests have been shifted to multi-processor systems. A list-scheduling based heuristic is proposed in [49], to dynamically recalculate the priority of communicating tasks. In [76], static and dynamic variable voltage scheduling heuristics for real-time heterogeneous embedded systems are proposed. An approach based on critical-path is used for selecting the voltage settings of tasks. However, both [49] and [76] assume that the task assignment is given. A similar problem to the one studied in this paper is investigated in [135]. A two-phase framework is presented to first determine the allocation of tasks onto processors and then the voltage settings of tasks using convex programming. In [138], a dynamic processor voltage adjustment mechanism for a homogeneous multi-processor environment is discussed. However, the time and energy costs for communication tasks are not addressed in any of [49], [135], and [138].

The goal of all the above works is to minimize the overall energy cost of the system. While such a goal is reasonable for tightly coupled systems, it does not capture the nature of WSNs. The reason is that to minimize the overall energy cost can lead to heavy use of energy-effective sensor nodes, regardless of their remaining energy. The consequent short lifetime of such sensor

nodes will very likely hinder the system from delivering required performance. This weakness is a major motivation of the proposed energy-balanced task allocation.

Our work considers the energy and time costs of both computation and communication tasks. As indicated by several researches, wireless communication is a major source of energy dissipation in WSNs. By incorporating techniques such as rate adaptation, we can greatly improve the energy-efficiency of the system.

Energy-balanced task allocation bears some resemblance to load-balance in distributed computing. However, the communication tasks over the same wireless channel need to be serialized such that run-time contentions can be avoided. The serialization imposes new challenges that distinguish our problem from most of the existing works for load-balance or real-time scheduling in distributed systems.

3.3 Problem Definition

3.3.1 System Model

Let v denote the number of sensor nodes. Let $NG =< V, L >$ denote the network graph for the cluster, where $V = \{V_i : i \in [v]\}$. Since we are considering collocated network, NG is a complete graph. We assume that all sensor nodes in V have the same processing and communication capabilities. Let K denote the number of communication channels that are of the same bandwidth. Let R_i denote the remaining energy of V_i, which can vary from each other.

We consider discrete voltage settings for sensor nodes. This is realistic as most commercial processors do not provide continuous voltage scaling. Specifically, each sensor node is equipped with d discrete voltage levels, denoted as $\{D_i : i \in [d]\}$ in decreasing order. As discussed in Section 2.2.1, for processing a specific task, each voltage level corresponds to a processing speed

in terms of the number of cycles per unit time. This processing speed in turn determines the processing delay and energy cost. More importantly, the energy cost is a convex and monotonically decreasing function of processing delay. Depending on the task size and instruction composition, such a function may vary for different tasks

We also consider discrete transmission rate for wireless communication. This is practical since the modulation setting in (2.4) is usually set to positive even integers in real systems. Specifically, the radio on each sensor node can transmit by choosing one level from a list of f modulation levels $\{F_i : i \in [f]\}$ in decreasing order. Based on our discussion in Section 2.2.2, for a given packet of fixed size, each rate level corresponds to a transmission delay and energy cost. For ease of presentation, we ignore the non-monotonicity discussed in Section 2.2.2. In other words, we assume that the output power always dominates the circuitry power. Hence, the energy cost for sending the packet is a convex and monotonically decreasing function of transmission delay, while the energy cost for receiving a packet is a linear function of transmission delay. These functions may vary for different links depending on the packet size.

In the above discussion, we have emphasized high level models that abstract the tradeoffs between energy and delay. Although technical details behind these model can be found in Section 2.2, it suffices to understand the problem and approach in this chapter simply based on the above high level models.

Regarding the *exclusive access constraint*, we assume that a non-preemptive scheduling policy is employed by each sensor node and each wireless channel. In other words, the time duration scheduled for different computation (communication) tasks over the same sensor node (wireless channel) cannot overlap with each other. Moreover, the underlying communication protocols are assumed to be capable of scheduling communication tasks according to the start time of each task in order to avoid run-time contentions.

For ease of analysis, we assume that the processors and radios are completely shut down in idle state. We note that to shut down the processor/radio whenever idle may not always save energy, since the energy for shutting down and restarting the radio can be larger than the energy cost for keeping the radio on during the idle state. Methods for determining the optimal working mode during idle state for energy savings is beyond the scope of this thesis and can be found in [109]. We also assume that ultra-low power paging or wakeup radios are available for waking up communication peers if necessary. Such radios are available on commercial sensor nodes, including Berkeley PicoRadio [136] and Sensoria products [102]. The time and energy cost of such radios are usually negligible.

3.3.2 Application Model

We consider a periodic application with latency constraint Γ for each execution of the application. We use the application graph $AG =< C, Q >$ to represent the intra-relationship between the computation and communication tasks, where $C = \{C_i : i \in [c]\}$ is the set of c computation tasks and $Q = \{Q_i : i \in [q]\}$ is the set of q communication tasks.

For most applications in WSNs, the source tasks are used for sensing or gathering raw data. Hence, it does not make sense to place more than one source task onto the same sensor node. We define a *task placement constraint* as that no two source tasks can be assigned to the same sensor node. Our model and approach can also be extended to handle the general case that any pair of tasks must be or must not be assigned to the same sensor node.

For each task $C_i \in C$, let W_i denote its workload in terms of the worst-case number of required computation cycles. Based on our discussion in Section 3.3.1, we can calculate the processing delay and energy cost for C_i at each voltage level. Let τ_{ij} denote the processing delay of C_i at the j-th voltage level, with ϵ_{ij} denoting the corresponding energy cost.

We assume that all communication tasks are performed by transmitting exactly one data packet with variable size. For each task $Q_i \in Q$, let weight s_i denote the size of the packets to be transmitted. Different edges incident from the same node may have different weights. For a communication task $Q_i = (j, k)$, if both C_j and C_k are assigned to the same sensor node, the time and energy cost of Q_i is zero. Otherwise, let τ'_{ij} denote the time cost of Q_i when the j-th modulation level is chosen for the transmission, with ϵ^s_{ij} denoting the corresponding sending energy and ϵ^r_{ij} denoting the corresponding receiving energy.

All the above values of τ_{ij}, ϵ_{ij}, τ'_{ij}, ϵ^s_{ij} and ϵ^r_{ij} can be calculated based on the above system and application models. Hence, we assume they are given a priori for our problem.

3.3.3 Task Allocation

Based on the above system and application models, a *task allocation* is defined as (1) the *assignment* of computation tasks onto sensor nodes and communication tasks onto channels, (2) the *voltage settings* of computation tasks, (3) the *rate setting* of communication tasks, and (4) the *scheduling* of computation and communication tasks. Each task can be assigned to exactly one sensor node with a fixed voltage setting. Also, each communication task can be assigned to exactly one channel with a fixed rate level. An allocation is feasible if it satisfies the latency, exclusive access, and task placement constraints.

The *system lifetime* is defined as the time duration from the time when the application starts execution to the time when any sensor node in the cluster fails due to depleted energy. A general solution to maximize the system lifetime is to allow variable task allocations in different periods. Consequently, the energy cost for each sensor node may vary in different periods. However, due to the high complexity raised by such a solution, we assume that the task allocation remains the same for all application periods. That is, the behavior of the system repeats for each period and every sensor node spends the same energy duration each period.

Let $\mathcal{E}_i$ denote the energy cost of $V_i \in V$ during each application period. Given an allocation, the system lifetime (in number of periods) can be calculated as $\min_i\{\lfloor \frac{R_i}{\mathcal{E}_i} \rfloor\}$. A feasible allocation is optimal if the corresponding system lifetime is maximized among all the feasible allocations.

Note that a more complex definition of the system lifetime would be the time period from the beginning of the application execution to the time when not enough sensor nodes are alive to deliver required performance. For example, it is shown that to perform the LOB algorithm at an acceptable accuracy requires at least three sensor nodes. However, such a definition is quite application-specific. Thus, a simple but general definition of the system lifetime is adopted in this thesis. Intuitively, to optimize the system lifetime with the above more complex definition, we may recursively apply the proposed optimization approaches to the resulting systems after sensor nodes die out. Now, our task allocation problem can be informally stated as:

Find an allocation of a set of communicating tasks onto a single-hop cluster that minimizes the maximal energy cost among all sensor nodes during each application period, normalized by their remaining energy.

3.4 Integer Linear Programming Formulation

In this section, we present an ILP formulation of our task allocation problem that captures the behavior of the system during one application period. We first list the notations used in the formulation in Table 3.1.

To capture the relative order imposed by the precedence constraints among tasks, we define the Constraint set 1 shown in Figure 3.1. It is easy to verify that the exclusive access constraint for tasks with precedence constraints is also enforced by Constraint set 1. However, for tasks that do not have precedence constraints between them, an extra set of constraints are needed (Constraint set 2 in Figure 3.2) to enforce the exclusive access constraint. In addition, the task placement constraint is captured by the Constraint set 3 in Figure 3.2.

Table 3.1: Table of notations for ILP formulation

Γ	latency constraint for each execution of the application
τ_{ij}, ϵ_{ij}	time and energy costs of executing task C_i using voltage level V_j
$\tau'_{ij}, \epsilon^s_{ij}$, ϵ^r_{ij}	time and energy costs of communication task $Q_i = (j, k)$, if C_j and C_k are not assigned to the same sensor node and the j-th modulation level is used for the transmission
$a \| b$	no dependency relationship exists for tasks a and b
$\{x_{ij}\}$	a set of 0-1 variables such that x_{ij} equals one iff C_i is assigned to V_j
$\{y_{ij}\}$	a set of 0-1 variables such that y_{ij} equals one iff the voltage of C_i is set to V_j
$\{z_{ij}\}$	a set of 0-1 variables such that z_{ij} equals one iff Q_i is assigned to the j-th channel
$\{u_{ij}\}$	a set of 0-1 variables such that u_{ij} equals one iff Q_i is transmitted at the j-th modulation level
$\{r_{ij}\}$	a set of 0-1 variables such that r_{ij} equals one iff C_i and C_j are assigned to the same sensor node
$\{s_{ij}\}$	a set of 0-1 variables such that s_{ij} equals one iff Q_i and Q_j are assigned to the same channel
$\{\alpha(i)\}$	a set of real variables indicating the time when C_i starts execution
$\{\beta(i)\}$	a set of real variables indicating the time when C_i completes execution
$\{\gamma(i)\}$	a set of real variables indicating the time when Q_i starts transmission
$\{\delta(i)\}$	a set of real variables indicating the time when Q_i completes transmission
$\{p_{ij}\}$	a set of 0-1 variables such that p_{ij} equals one iff the execution of C_i finishes before C_j starts
$\{t_{ij}\}$	a set of 0-1 variables such that t_{ij} equals one iff the transmission of Q_i finishes before Q_j starts

The complete ILP formulation is given in Figure 3.3, where $\mathcal{E}$ is an auxiliary variable. In the figure, the term $\sum_{C_i \in C}\{x_{ik} \sum_j (y_{ij}\epsilon_{ij})\}$ gives the energy cost for computation tasks on V_k. The term $\sum_{Q_i=(a,b)\in Q}\{x_{ak}(1-x_{bk}) \sum_j (u_{ij}\epsilon^s_{ij}) + (1-x_{ak})x_{bk} \sum_j (u_{ij}\epsilon^r_{ij})\}$ gives the energy costs for all the communication tasks that involve V_k.

Clearly, the presented formulation is non-linear. It can be transformed into an ILP formulation by standard linearization techniques [122]. We omit the details of linearization in this thesis.

3.5 Heuristic Approach

In this section, we describe an efficient 3-phase heuristic for solving the task allocation problem. Initially, we set the voltage and rate levels for all tasks to the highest option. In the first phase,

Constraint set 1:

$\forall C_i \in C$

$\sum_j x_{ij} = 1$	// every task can be assigned to exactly one sensor node
$\sum_j y_{ij} = 1$	// every task can be executed using exactly one voltage level
$\alpha(i) \geq \max_{Q_l=(j,i)\in Q}\{\delta(l)\}$	// C_i starts execution after receiving all inputs
$\beta(i) = \alpha(i) + \sum_j (y_{ij}\tau_{ij})$	// execution time of C_i depends on the voltage level

$\forall C_i, C_j \in C$

$r_{ij} = 1$ iff $\forall k = 1, \ldots, v,\ x_{ik} = x_{jk}$	// r_{ij} equals one if C_i and C_j are assigned to the same sensor node

$\forall Q_i = (a, b) \in Q$

$\sum_j z_{ij} = 1$	// Q_i can be assigned to exactly one channel
$\sum_j u_{ij} = 1$	// Q_i can be transmitted at exactly one rate
$\gamma(i) \geq \beta(a)$	// Q_i starts transmission after C_a finishes
$\delta(i) = \gamma(i) + \sum_j (u_{ij}\tau'_{ij})(1 - r_{ab})$	// the transmission time of Q_i depends on the locations of C_a and C_b and its rate

for any source tasks C_i

$\alpha(i) \geq 0$	// all source tasks can start execution at time 0

for any sink task C_i

$\beta(i) \leq \Gamma$	// all sink tasks must complete within the latency constraint

Figure 3.1: Constraint sets 1 for the ILP formulation

the tasks are grouped into clusters with the goal to minimize the overall execution time of the application. In the second phase, task clusters are assigned to sensor nodes such that the highest energy cost among all sensor nodes, normalized by their remaining energy, is minimized. In the last phase, the system lifetime is maximized by lowering the voltage levels of tasks. The details of the heuristic are as follows.

Phase 1: A *task cluster* is defined as a set of tasks assigned to the same sensor node with a specific execution order. Communication between tasks within a cluster costs zero time and energy. In this phase, we assume an unlimited number of sensor nodes, implying that the number of clusters is also unlimited. The main purpose of this phase is to eliminate communication tasks in order to reduce the overall execution time of the application.

Constraint set 2:

$\forall C_i, C_j \in C$, such that $i \neq j$ and $C_i || C_j$

$p_{ij} = 1 - p_{ji}$ // p_{ij} is the inverse of p_{ji}

$\alpha(j) \geq p_{ij} r_{ij} \beta(i)$ // if C_i and C_j are assigned to the same node, // C_i completes before C_j starts execution // iff $p_{ij} = 1$

$\forall Q_i, Q_j \in Q$, such that $Q_i = (a, b)$, // Communiation tasks from or to the same
$Q_j = (a', b')$, either $a = a'$ or $b = b'$ // computation task

$t_{ij} = 1 - t_{ji}$ // t_{ij} is the inverse of t_{ji}

$\gamma(j) \geq t_{ij}(1 - r_{ab})(1 - r_{a'b'})\delta(i)$ // Q_i completes before Q_j starts transmission // iff $t_{ij} = 1$

$\forall Q_i, Q_j \in E$, such that $Q_i = (a, b)$, // Communication tasks between two different
$Q_j = (a', b')$, $a \neq a'$, $b \neq b'$, and $Q_i || Q_j$ // pair of computation tasks

$t_{ij} = 1 - t_{ji}$ // t_{ij} is the inverse of t_{ji}

$s_{ij} = 1$ iff $\forall k = 1, \ldots, K$, $z_{ik} = z_{jk}$ // s_{ij} equals one if Q_i and Q_j are assigned to // the same channel

$\gamma(j) \geq t_{ij}(1 - r_{ab})(1 - r_{a'b'})s_{ij}\delta(i)$ // if Q_i and Q_j are assigned to the same channel, // Q_i completes before Q_j starts transmission // iff $t_{ij} = 1$

Constraint set 3:

$\forall C_i, C_j \in C$, such that C_i and C_j are source tasks and $i \neq j$

$r_{ij} = 0$ // any two source tasks cannot be assigned to the same sensor node

Figure 3.2: Constraint sets 2 and 3 for the ILP formulation

The idea of Phase 1 is similar to the algorithm proposed in [96] (pp. 123 - 131). However, traditional approaches for task clustering usually assume a full connection among processors such that communication tasks can be parallelized, whereas in our problem, communication tasks over the same channel must be serialized. Thus, a new challenge is to select a policy for the serialization that facilitates the reduction of the execution time of the application. We use a simple first-come-first-serve policy to order the communication tasks ready at different

Minimize $\mathcal{E}$

Subject to $\forall V_k \in V$ $\quad \frac{\sum_{C_i \in C}\{x_{ik}\sum_j(y_{ij}\epsilon_{ij})\}}{R_k} + \frac{\sum_{Q_i=(a,b)\in Q}\{x_{ak}(1-x_{bk})\sum_j(u_{ij}\epsilon^s_{ij}) + (1-x_{ak})x_{bk}\sum_j(u_{ij}\epsilon^r_{ij})\}}{R_k} \leq \mathcal{E}$

and Constraint sets 1, 2, and 3

Figure 3.3: ILP formulation for the energy-balanced task allocation problem

times. Communication tasks ready at the same time (such as those initiated by the same task) are executed in a non-decreasing order of their communication loads. Nevertheless, more sophisticated policies are also applicable.

---

1. Each task is assumed to constitute a cluster by itself
2. Set Q as the list of communication tasks in a non-decreasing order of their weights
3. $\Phi \leftarrow$ Travese()
4. **While** Q is not empty **Do**
5. Remove the first edge from Q, denoted as (i, j)
6. $\Phi' \leftarrow$ Traverse() as if $CL(i)$ and $CL(j)$ are merged
7. **If** $\Phi' < \Phi$ and to merge $CL(i)$ and $CL(j)$ does not violate the task placement constraint
8. Merge $CL(i)$ and $CL(j)$
9. $\Phi \leftarrow \Phi'$
10. **If** $\Phi > \Gamma$, **Return** failure

Figure 3.4: Pseudo code for Phase 1

The pseudo code for Phase 1 is shown in Figure 3.4. In the code, Φ denotes the overall execution time of the application and $CL(i)$ denotes the cluster that contains task C_i. Initially, every task is assumed to constitute a cluster by itself. We then examine all the edges in a non-increasing order of their weights. For each edge, (i, j), if the execution time of the application can be reduced by merging $CL(i)$ with $CL(j)$ without violating the task placement constraint, we perform the merge. Otherwise, C_i and C_j remain in two different clusters. In lines 3 and 6, the function Traverse() is called to traverse the DAG in order to determine the schedule of the tasks and hence Φ.

The pseudo code for Traverse() is shown in Figure 3.5. In the code, we maintain a queue of tasks, A, that stores all the ready computation or communication tasks in their expected execution order. We also maintain a timestamp for each task cluster that indicates the finish time for all scheduled tasks within the cluster. Similarly, we maintain a timestamp for each channel that indicates its nearest available time. The timestamps are used to schedule the computation and communication tasks in lines 7, 13, and 14. In lines 9 and 14, the timestamps

are updated based on the execution time of the scheduled tasks. The actions in lines 17 and 18 are to ensure that the radio can be tuned to at most one channel at any time.

---

1. Initialize A
2. Set the timestamps for all task clusters and channels to zero
3. Append all source tasks to A with ready time set to zero
4. **While** A is not empty **Do**
5. Remove the first task from A
6. **If** the removed task is a computation task, denoted as C_i
7. Set $\alpha(i) \leftarrow \max\{\text{ready time of } C_i, \text{timestamp of } CL(i)\}$
8. Set $\beta(i)$ to the expected completion time of C_i, i.e., $\beta(i) \leftarrow \alpha(i) + \tau_{i1}$
9. Set the timestamp of $CL(i)$ to $\beta(i)$
10. Insert all communication tasks initiated by C_i into A with ready time set to $\beta(i)$, in a non-decreasing order of their communication loads
11. **Else**
12. Let $Q_i = (a, b)$ denote the removed communication task
13. Find the channel with the smallest timestamp, say the j-th channel
14. Set $\gamma(i) \leftarrow \max\{\text{ready time of } Q_i, \text{timestamp of the } j\text{-th channel}\}$
15. Set $\delta(i)$ to the expected completion time of Q_i, i.e., $\delta(i) \leftarrow \gamma(i) + \tau'_{i1}$
16. Set the timestamp of the j-th channel to $\delta(i)$
17. Set the ready time of any unscheduled communication tasks from C_a to $\delta(i)$
18. Set the ready time of any unscheduled communication tasks to C_b to $\delta(i)$
19. **If** all the communication tasks to C_b have been scheduled
20. Insert C_b into A with ready time set to $\delta(i)$
21. **Return** the largest timestamp among all clusters

Figure 3.5: Pseudo code for function Traverse()

Phase 2: In this phase, we assign the task clusters from Phase 1 onto the actual sensor nodes in V. Note that multiple clusters can be assigned to the same sensor node. Based on the contained tasks and the corresponding communication tasks, we first calculate the energy cost of each cluster. Let m denote the number of task clusters obtained from Phase 1. Let $\pi = \{\pi_i : i \in [m]\}$ denote the list of all tasks clusters and ξ_i denote the energy cost of π_i. The normalized energy cost (*norm-energy* for short) of a sensor node is given as the sum of the energy cost of the clusters assigned to the sensor node, normalized by the remaining energy of the sensor node.

The pseudo code of Phase 2 is shown in Figure 3.6. Initially, π is sorted into a non-increasing order of energy cost of clusters. Then, for each cluster in π, we calculated the norm-energy of

every sensor node as if the cluster is assigned to the sensor node (called *expected norm-energy*). We then assign the cluster to the sensor node that gives the minimal expected norm-energy. In the code, function TraverseAssigned() is used to find the execution time of the application based on the resulting assignment. Compared with Traverse(), the modification in TraverseAssigned() is that in line 7 of Figure 3.5, each computation task is scheduled on the sensor node that it is assigned to. Thus, timestamps are maintained for all sensor nodes, instead of task clusters.

---

1. Sort π in a non-increasing order of the energy cost of clusters
2. **While** π is not empty **Do**
3. Choose the first element, π_1, in π
4. Calculate the expected norm-energy for each sensor node (set to infinity if two source tasks are assigned to the same sensor node)
5. Assign π_1 to the sensor node that gives the minimal expected norm-energy
6. Update the norm-energy of the sensor node
7. Remove π_1 from π
8. $\Phi \leftarrow$ TraverseAssigned()
9. **If** $\Phi > \Gamma$, **Return** failure

Figure 3.6: Pseudo code for Phase 2

Phase 3: The voltage levels of computation tasks and the rate level of communication tasks are adjusted in this phase with the goal to maximize the system lifetime. An iterative greedy heuristic is used (shown in Figure 3.7). Let $\mathcal{E}$ denote the maximum of the norm-energy among all sensor nodes. The sensor node that determines $\mathcal{E}$ is called the *critical node*. In each iteration, we find the task such that by lowering its current voltage level to the next level, $\mathcal{E}$ can be decreased the most. The increased latency caused by lowering the voltage or rate level is added to Φ. Since the schedule of tasks can be changed by the latency increment, Φ is re-computed by traversing the DAG every time it reaches Γ (in line 15).

To do so, for each $V_i \in V$, we create a list of computation and communication tasks that involve V_i, denoted as T_i. For each task in $t_j \in T_i$, we also associate two quantities to t_j that indicate the reduction in energy cost and increment in latency if the voltage or transmission

rate of t_j is lowered to the next level. Let ed_j denote the reduction in energy and td_i denote the increment in latency. Also, let ED_i denote the list of ed_j's for all tasks in T_i.

---

1. For each V_i, sort ED_i in a non-increasing order
2. **Do**
3. $i \leftarrow 1$
4. Let V_r denote the critical sensor node
5. **While** $i \leq |ED_r|$ **Do**
6. Select the i-th component in ED_r; let a denote the corresponding task in T_r
7. **If** $\Phi + td_a > \Gamma$, $i = i + 1$
8. **Else**
9. $\Phi \leftarrow \Phi + td_a$
10. **If** a is a computatin task
11. Lower the voltage level of a to the next available option
12. **Else**
13. **If** to lower the modulation level of a to the next available option does not increase $\mathcal{E}$
14. Lower the modulation level of a to the next available option
15. **Else** $i \leftarrow i + 1$
16. **If** any voltage or modulation scaling is performed
17. Update ed_a and td_a; resort ED_r if necessary
18. Find the new critical sensor node, $V_{r'}$; update $\mathcal{E}$
19. **If** $r \neq r'$
20. $r \leftarrow r'$; $i \leftarrow 1$
21. $\Phi \leftarrow$ TraverseAssigned()
22. **Until** $\mathcal{E}$ can not be reduced any more

Figure 3.7: Pseudo code for Phase 3

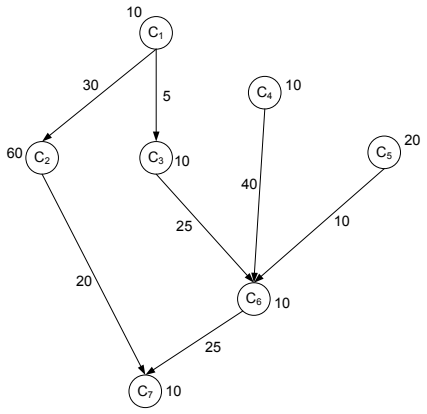
One concern is that to decrease the transmission energy at the sender, we actually increase the receiving energy at the receiver. Thus, in lines 13 and 14 of Figure 3.7, we ensure that the modulation scaling is performed only when the increase in the reception energy does not cause the value of $\mathcal{E}$ to increase. By doing so, our heuristic can handle the situation in highly dense WSNs, where the receiving energy is comparable with the sending energy.

Time Complexity Analysis: In Phase 1 (Figure 3.4), the **While** iteration is executed q times (recall that q is the number of communication tasks). Function Traverse() in line 6 takes $O(c+q)$ time (recall that c is the number of computation tasks). Thus, Phase 1 needs $O(q(c+q))$ time. In Phase 2 (Figure 3.6), the ordering in line 1 takes $O(m \log m)$ time (recall that m is

the number of clusters obtained after Phase 1). The outer iteration is executed m times. The results of v possible assignments are compared in line 5 (recall that v is the number of sensor nodes). The traverse in line 8 takes $O(c+q)$ time. Hence, Phase 2 takes $O(m \log m + vm + c + q)$ time. In Phase 3 (Figure 3.7), the sorting in line 1 takes $O((c+q)\log(c+q))$ time. The number of voltage switching in line 11 is bounded by dc (recall that d is the number of voltage options). The number of rate switching in line 14 is bounded by fq (recall that f is the number of available rate options). To update ED_r in line 10 needs $O(\log(c+q))$ time. Let ϕ denote the number of times for calling TraverseAssigned() in line 15. The time complexity of Phase 3 is $O((c+q)\log(c+q) + (dc+fq)\log(c+q) + \phi(c+q)) = O(dc+fq)\log(c+q) + \phi(c+q)$. Although ϕ equals $dc+fq$ in the worst case, it was observed in our simulations that ϕ is usually less than 5.

Thus, the overall time complexity of the heuristic is $O(q(c+q) + (m \log m + vm + c + q) + (dc + fq)\log(c+q) + p(c+q)$. Since $m \leq c$ and $\psi \leq dc + fq$, the worst case time complexity can be simplified as $O((dc+fq)(c+q+\log(c+q)) + (q+v)c)$ in the worst case. Assuming that the number of voltage and rate options are fixed for given sensor node hardware, the running time scales quadratically with number of tasks and linear with number of sensor nodes.

An Illustrative Example: We illustrate the execution of the above heuristic through a simple example. We assume a cluster of 3 sensor nodes connected by 2 channels. Each sensor node have two voltage levels, D_h and D_l, with the speed for D_h equal to 1 and the speed for D_l equal to 0.3. We assume that rate adaptation is not available and it costs one time and energy unit for transmitting one data unit over any channel. The application graph is shown in Figure 3.8(a), with each circle representing a task. The number close to each circle is the required workload, while the number on each edge is the weight of the edge. The time and energy costs for executing tasks at the two voltage levels are given in Figure 3.8(b). We assume that $\Gamma = 250$ time units.



(a) application graph

task	time cost		energy cost	
	D_h	D_l	D_h	D_l
C_1	10	33	20	6
C_2	60	199	120	36
C_3	10	33	20	6
C_4	10	33	20	6
C_5	20	66	40	12
C_6	10	33	20	6
C_7	10	33	20	6

(b) time and energy costs for executing tasks at voltage levels D_h and D_l

Figure 3.8: An application example

The clustering steps in Phase 1 is shown in Figure 3.9. In this phase, the voltage levels of all tasks are set to D_h. The sorted edge list with respect to edge weights is $\{(C_4, C_6), (C_1, C_2), (C_3, C_6), (C_6, C_7), (C_2, C_7), (C_5, C_6), (C_1, C_3)\}$. The table in Table 3.2 traces the execution of the algorithm, where Φ_i is the execution time of the application at the completion of step i. The sub-figures (a) through (e) in Figure 3.9 correspond to the application graph at the completion of steps 0, 1, 2, 3, and 5, respectively. The clusters are marked with polygons in dash line. Note that in steps 6 and 7, the clustering is not performed due to the task placement constraint.

Table 3.2: Trace of clustering steps in Figure 3.9

step i	edge examined	L if clustering	clustering?	Φ_i
0				145
1	(C_4, C_6)	135	yes	135
2	(C_1, C_2)	120	yes	120
3	(C_3, C_6)	100	yes	100
4	(C_6, C_7)	100	no	100
5	(C_2, C_7)	80	yes	80
6	(C_5, C_6)		no	80
7	(C_1, C_3)		no	80

During Phase 2, we first calculate the energy costs for each cluster – 190 energy units for cluster $\pi_1 = \{C_1, C_2, C_7\}$, 100 for the cluster $\pi_2 = \{C_3, C_4, C_6\}$, and 50 for cluster $\pi_3 = \{C_5\}$.

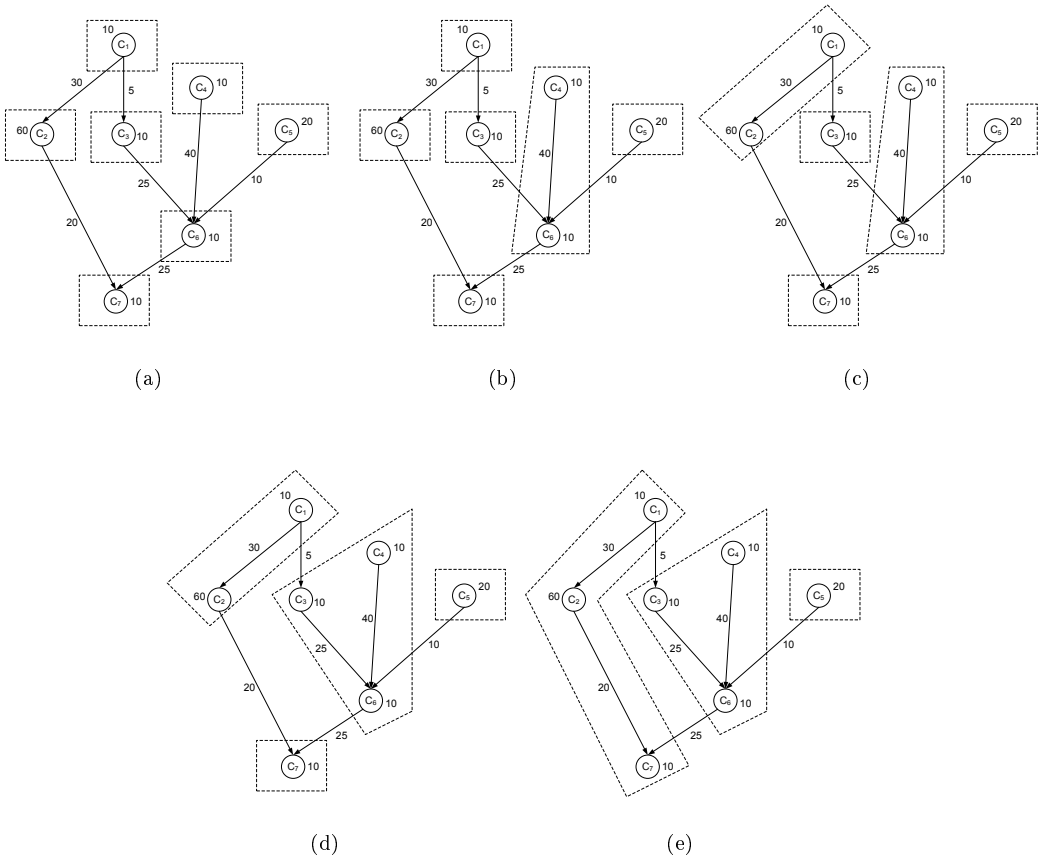


Figure 3.9: Clustering steps for the application in Figure 3.8

Since the remaining energy for the three sensor nodes are the same, we simply assign π_1 to V_1, π_2 to V_2, and π_3 to V_3.

Finally, we adjust the voltage levels of tasks. Since V_1 is the critical node, we first set the voltage level of C_2 to V_l, which reduces $\mathcal{E}_1$ to 106 and increases Φ from 80 to 219. Next, we set the voltage level of C_1 to V_l, which further decreases $\mathcal{E}_1$ to 92 and increases Φ to 242. After this step, the critical node becomes V_2 with $\mathcal{E}_2 = 0.1$. Since the latency constraint is 250, our heuristic terminates.

In the above example, we decreases the norm-energy of the critical sensor node from 0.19 to 0.1, implying a system lifetime improvement by a factor around 2.

3.6 Experimental Results

A simulator based on the system and application models presented in Section 3.3 was developed to evaluate the performance of our approach using application graphs from both a synthetic approach and real world problems. The goals of our simulations are (1) to measure and compare the performance of the 3-phase heuristic against the ILP-based approach; and (2) to evaluate the impact of the variations in several key system parameters on the performance of the heuristic, including the tightness of the latency constraint, the relative time and energy costs of communication tasks compared with computation tasks, and the number of voltage levels.

The evaluation metrics are based on the system lifetime obtained by different approaches. Let LT_{ILP} and LT_{heu} denote the system lifetime obtained by the ILP-based approach and the 3-phase heuristic, respectively. In addition, let LT_{raw} denote the system lifetime obtained by assuming that no voltage or modulation scaling is available (i.e., every sensor node runs and transmits data at the highest speed). Since we do not have a stand alone approach to obtain LT_{raw}, LT_{raw} was calculated based on the value of $\mathcal{E}$ obtained after phase 2 of the 3-phase heuristic.

Unless otherwise stated, all the data presented in this section is averaged over more than 100 instances so that a 95% confidence interval with a 10% (or better) precision is achieved.

3.6.1 Synthetic Application Graphs

We first show a set of results where only voltage scaling is considered. This is followed by results that incorporate rate adaptation.

Experimental Procedure: The structure of the application graph was generated using a method similar to the one described in [35]. The only difference is that we enforce multiple source tasks in the generation of the DAG.

According to Rockwell's WINS node [123], the power consumption of an Intel StrongARM 1100 processor with 150 MIPS is around 200 mW. This implies that the time and energy costs per instruction are around 5 nSec and 1 nJ. Also, the power of the radio module used in WINS is 100 mW at 100 Kbps, implying that the time and energy costs for transmitting a bit are around 10 μSec and 1 μJ. In the following, we set the parameters for our simulator such that the time and energy costs for computation and communication tasks roughly follow the above numbers.

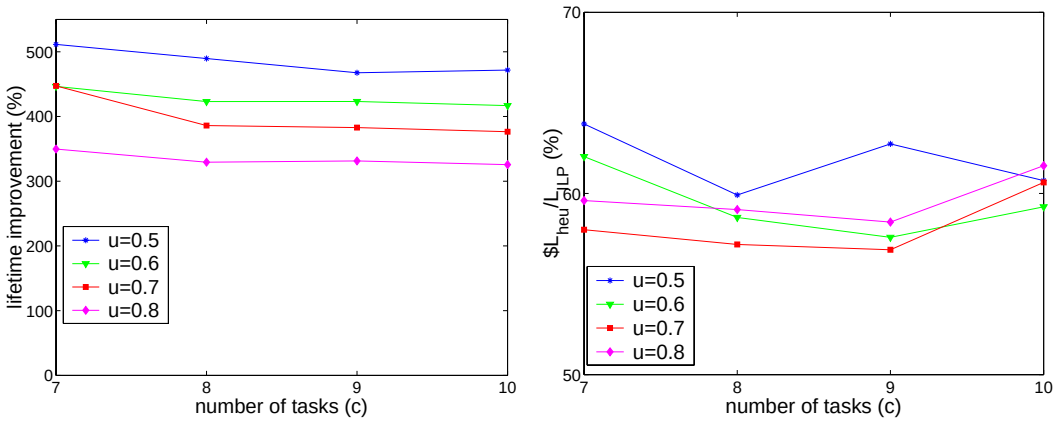
We set the maximum computation speed of each sensor node to 10^2 Mcps (million cycles per second) and the minimum speed to 0.3×10^2 Mcps. It is assumed that other levels of computation speed are uniformly distributed between the maximum and minimum speeds. The computation requirements of the tasks followed a gamma distribution with a mean value of 2×10^5 and a standard deviation of 10^5. From [24, 80], the power consumption of the processor can be modeled as a polynomial function of processing speed, $g(SP)$ of at least degree 2, where SP is the speed of the processor. Hence, we set the power function of task C_i to be $a_i \cdot (\frac{SP}{10^8})^{b_i}$, where a_i and b_i were random variables with uniform distribution between 2 and 10, and 2 and 3 [80], respectively. For example, suppose $a_i = b_i = 2$. Then, to execute a task of 2×10^5 instructions costs 2 mSec and 4 mJ in the highest speed, and 6.7 mSec and 1.2 mJ in the lowest speed.

The time and energy costs of communication tasks are determined by the number of data units to transmit and the values of τ and ε. Based on the radio characteristics for WINS, we assume that the time for transmitting one bit is 10 μSec and the corresponding energy cost is 1 μJ. To focus on the main issues, we set the startup energy cost of the radio to be zero. To study the effect of different communication load with respect to the computation load, the number of bits per communication task follows a uniform distribution between $200CCR(1 \pm 0.2)$, where CCR (communication to computation ratio) is a parameter indicating the ratio of the average

execution time of the communication tasks to that of the computation tasks. Intuitively, a larger value of CCR implies a relatively heavier communication loads compared with the computation loads. Note that by varying CCR, we abstract not only the variations in the amount of transmitted data, but also the variations in the relative speed of computation and communication devices. In our simulations, CCR was varied within $[0, 20]$. In the above distribution, the coefficient is set to 200 so that when $CCR = 1.0$, the average time for executing either a task or a communication activity equals 2 mSec, which is also the average execution time for a computation task at the highest processing speed.

The period of the application, Γ, was generated in the following way. We first define the *distance* of a node in the application DAG as the number of edges in the longest path from the source to the node. Nodes are then divided into layers, with nodes in each layer having the same value of distance. Since the average time to execute a task in the highest speed is 2 mSec, we estimate the the computation time required for a layer with κ computation tasks as $2\lceil\frac{\kappa}{v}\rceil$ mSec, where v is the number of sensor nodes in the cluster. By doing so, we implicitly assume full parallelism in executing the tasks at each layer. In addition, the expected number of communication tasks initiated by a task is estimated as its out-degree subtracted by 1. Since the average time cost for a communication task is 2CCR mSec, we estimate the communication time required for a layer with totally η communication tasks as $2CCR\lceil\frac{\eta}{f}\rceil$ mSec, where f is the number of channels. Γ is then set to the sum of the computation and communication time cost of all layers over u, where $u \in [0, 1]$ is a parameter that approximates the overall utilization of the system. The setting of u is important as it determines the latency laxity for trading against energy. Intuitively, a larger value of u implies a tighter latency constraint and hence less latency laxity.

The remaining energy of sensor nodes follows a uniform distribution between $E_{mean}(1 \pm 0.3)$, where E_{mean} is a fairly large number.



(a) lifetime improvement achieved by the ILP-based approach

(b) Performance comparison of the ILP-based approach and the 3-phase heuristic

Figure 3.10: Lifetime improvement of our approaches for small scale problems (3 sensor nodes, 3 voltage levels, 2 channels, CCR = 1)

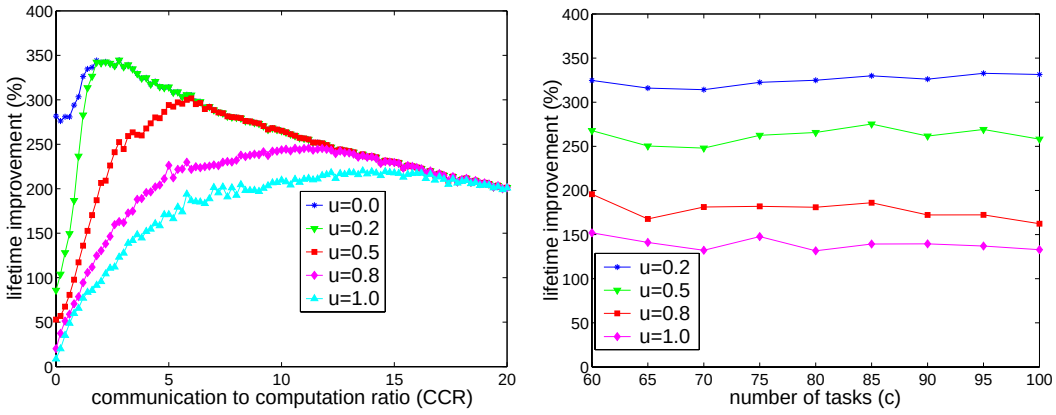
Small Scale Problems: We first conducted simulations for small scale problems, with 3 sensor nodes, 3 voltage levels, 2 channel, and 7 - 10 tasks. The number of source tasks in the application graph is set to 2, while the maximal in-degree and out-degree for each node are set to 3. A commercial software package, LINDO [72], was used to solve the ILP problems. Due to the large running time for solving some problem instances, LINDO was interrupted after two hours of execution if the optimal solution was not yet found. Then, the best solution obtained so far was returned. We observed that in most cases, LINDO was able to find the optimal solution within two hours.

The data shown in Figure 3.10 is averaged over more than 70 instances so that each data point has a 95% confidence interval with a 10% precision. In Figure 3.10(a), we illustrate the lifetime improvement achieved by the ILP-based approach, which is calculated as $\frac{LT_{ILP}}{LT_{raw}} - 1$. We can see an improvement around 3x - 5x. Figure 3.10(b) shows the performance ratio of the 3-phase heuristic over the ILP-based approach, i.e., $\frac{LT_{heu}}{LT_{ILP}}$. We can see that the 3-phase heuristic

achieved up to 63% of the solution obtained by the ILP-based approach for the conducted simulations.

While the running time of the heuristic is around zero, the average running time of the ILP-based approach ranges from 550 Sec ($c = 7$, $u = 0.5$) to 5900 Sec ($c = 10$, $u = 0.8$) on a Sun Blade1000 machine with a UltraSparc III 750 Mhz CPU.

Large Scale Problems: A set of simulations were conducted for evaluating the performance of the 3-phase heuristic for problems with 10 sensor nodes, 8 voltage levels, 4 channels, 60 - 100 tasks, $CCR \in [0, 20]$, and $u \in [0, 1]$. The number of source tasks in the application graph is set to 6. The maximal in-degree and out-degree for each node are set to 5. Due to the large size of the problems, it is impractical to obtain the optimal solutions by using the ILP-based approach. Thus, we use the lifetime improvement achieved by the 3-phase heuristic as the evaluation metric, which is calculated as $\frac{LT_{heu}}{LT_{raw}} - 1$. The simulation results are shown in Figure 3.11.



(a) lifetime improvement vs. system utilization (u) and communication to computation ratio (CCR)

(b) lifetime improvement vs. number of tasks ($CCR = 4$)

Figure 3.11: Lifetime improvement of the 3-phase heuristic for large scale problems (10 sensor nodes, 8 voltage levels, 4 channels, 60-100 tasks)

An improvement up to 3.5x in the system lifetime can be observed from Figure 3.11(a). We can see that the improvement increases when u decreases, as the latency laxity increases accordingly. The lifetime improvement saturates when u approaches 0, i.e., the latency constraint approaches ∞. Hence, the curve with $u = 0.0$ gives the upper bound of the improvement that can be achieved by our heuristic with respect to variations in CCR.

The effect of CCR is more complicated. For example, when $u = 0.5$, the lifetime improvement increases when $CCR \leq 6$ and decreases when CCR is beyond 6. This is because when CCR is small, the computation tasks dominate the overall energy costs of the application. By increasing CCR, we actually increase the latency constraint without increasing the computation load, which in turn can be traded for lifetime improvement. However, when CCR reaches some threshold value, the communication energy cost becomes more significant than that of the computation tasks. Thus, the lifetime improvement achieved by reducing computation energy becomes limited. We shall see later that this shortcoming can be overcome by incorporating rate adaptation into our heuristic.

Figure 3.11(b) shows the lifetime improvement with number of tasks, c varying from 60 to 100. We can see that the performance of our approach is quite stable with respect to the variation in c.

The miss rate (defined as the ratio of the number of instances that an approach fails to find a feasible solution to the total number of instances) of a heuristic is another key issue. Note that in our simulations, not all instances are guaranteed to have feasible solutions. We observed that the miss rate of the 3-phase heuristic is significant only when CCR is close to zero. Thus, we show the miss rate with $CCR = 0$ in Figure 3.12. Also, the running time of the heuristic is around 0.5 mSec on a Sun Blade1000 machine with a UltraSparc III 750 Mhz CPU.

Impact of the Number of Voltage Levels: We also studied the impact of the variations in the number of voltage levels. Simulations were conducted with 10 sensor nodes, 60 tasks,

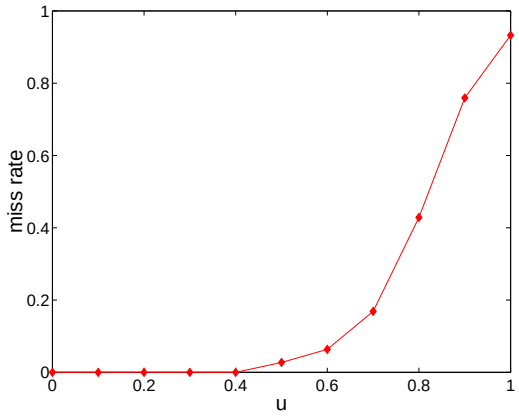


Figure 3.12: Miss rate of the 3-phase heuristic (10 sensor nodes, 8 voltage levels, 4 channels, 60 tasks, $CCR = 0$)

4 channels, $CCR = 2$, $u \in \{0.2, 0.5, 0.8, 1.0\}$ and 1 to 10 voltage levels. The results are demonstrated in Figure 3.13.

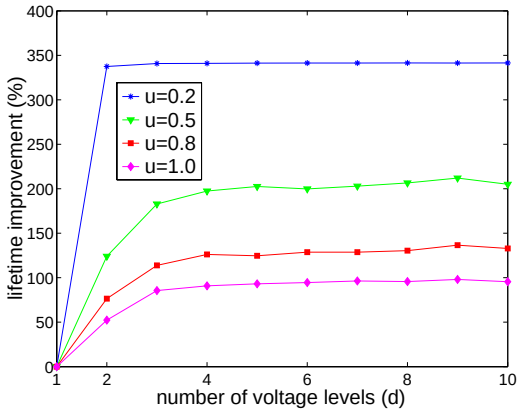


Figure 3.13: Impact of variation in number of voltage levels (10 sensor nodes, 4 channels, 60 tasks, $CCR = 2$)

The plots show that when $u > 0.2$, the performance of the heuristic can be significantly improved by increasing the number of voltage levels from 1 to 4. Further increase in the number of voltage levels does not improve the performance much. This is understandable since the energy behaves as a monotonically increasing and strictly convex function of the computation speed. The first derivative of the energy function tends to ∞ when the speed tends to ∞. Thus,

the most portion of energy saving is obtained by changing the speed from the highest option to some lower options, which can be efficiently achieved with 4 voltage levels per sensor node.

When $u = 0.2$, the latency laxity is so large that the voltage level of most tasks can be set to the lowest option. Thus, there is almost no improvement by increasing the number of voltage levels beyond 2.

Incorporating Rate Adaptation: We used the energy model of modulation scaling in Section 2.2.2 to illustrate the incorporation of rate adaptation. Due to the underlying single-hop connection, we assume that all sensor nodes have the identical settings for parameters C_{tr}, C_{ele}, and R_S. From [98], we set $C_{ele} = 10^{-7}$. To investigate the impact of different energy/time ratio for data transmission, we set C_{tr} to 10^{-7} and 10^{-6} for different instances. The modulation level, b, was set to even numbers between 2 and 6. We set $R_s = 1.7 * 10^4$ so that when $b = 6$, it roughly takes 10 μSec and 1 μJ to transmit a bit (as shown in Figure 3.14).

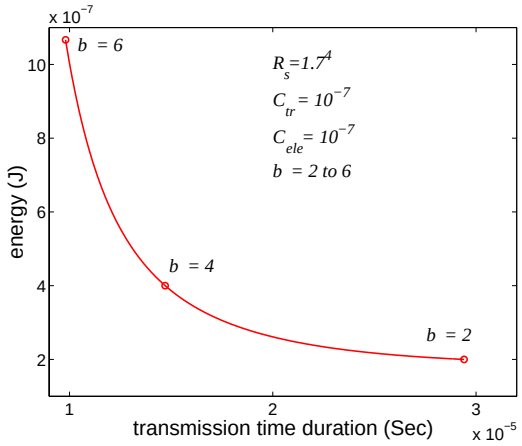
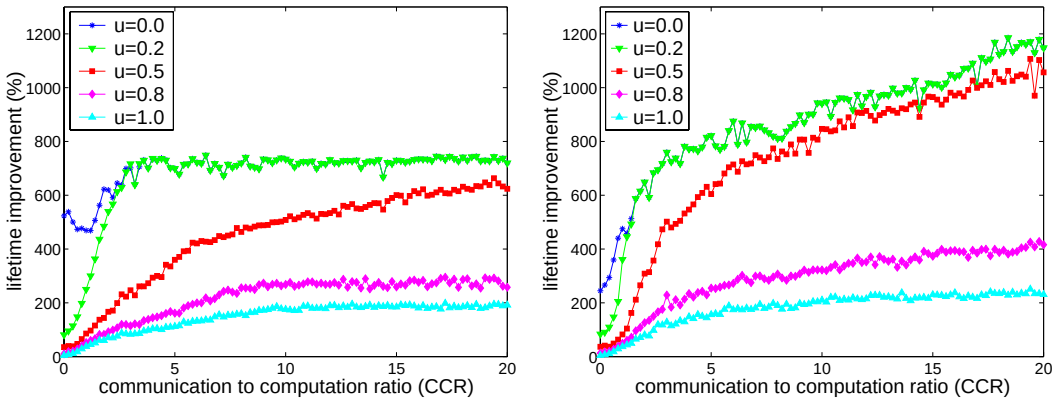


Figure 3.14: Energy-latency tradeoffs for transmitting one bit of data

The simulations were conducted with 10 sensor nodes, 8 voltage levels, 3 modulation levels ($\{2, 4, 6\}$), 60 tasks, $u \in \{0.0, 0.2, 0.5, 0.8, 1.0\}$, and $CCR \in [0, 20]$. Compared with Figure 3.11, we can observe a significant amount of performance improvement in Figure 3.15. For example, when $u = 0.5$, the highest lifetime improvement increases from 3x in Figure 3.11(a) to 6x in Figure 3.15(a) and even 10x in Figure 3.15(b). The difference in performance improvement



(a) small energy/time ratio for communication tasks ($C_{tr} = 10^{-7}$)

(b) large energy/time ratio for communication tasks ($C_{tr} = 10^{-6}$)

Figure 3.15: Lifetime improvement of the 3-phase heuristic incorporated with modulation scaling (10 sensor nodes, 8 voltage levels, 4 channels, 3 modulation levels, 60 tasks)

of Figures 3.15(a) and 3.15(b) is because that a larger C_{tr} leads to larger energy/time ratio of communication tasks, which in turn gives more advantage in reducing the communication energy by utilizing modulation scaling.

Similar to Figure 3.11, larger improvement is observed when u becomes smaller. In addition, the miss rate of the heuristic exhibits a similar trend as the cases with voltage scaling only.

3.6.2 Application Graphs from Real World Problems

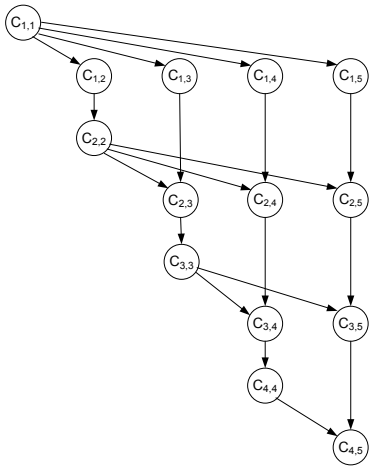
In addition to synthetic application graphs, we also considered application graphs of two real world problems: LU factorization algorithm [28] and Fast Fourier Transformation [29]. These two algorithms are widely used as kernel operations for various signal processing, such as beamforming [84].

LU Factorization: Figure 3.16(a) gives the sequential program for the LU factorization without pivoting, where s denotes the dimension of the matrix. The application graph of the algorithm for the special case of $s = 5$ is given in Figure 3.16(b). Each $C_{k,k}$ represents a pivot column operation and each $C_{k,j}$ represents an update operation. The total number of tasks

in the application graph equals $\frac{s^2+s-2}{2}$. Also, we assume the input matrix is available at the sensor node where task $C_{1,1}$ is assigned.

LU-Factorization(a)
1. **For** $k = 1$ to $s-1$ **Do**
2. **For** $i = k+1$ to s **Do** // $T_{k,k}$
3. $a_{ik} = a_{ik}/a_{kk}$
4. **For** $j = k+1$ to s **Do**
5. **For** $i = k+1$ to s **Do** // $T_{k,j}$
6. $a_{ij} = a_{ij} - a_{ik}/a_{kj}$

(a) sequential algorithm



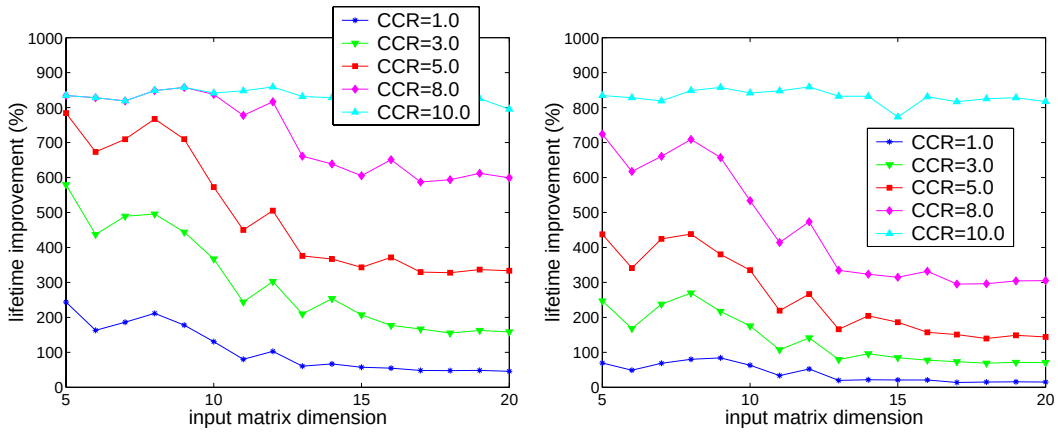
(b) example application graph with a 4×4 matrix

Figure 3.16: Matrix factorization algorithm

We performed simulations with 10 sensor nodes, 8 voltage levels, 4 channels, 3 modulation levels, and the matrix dimension, s, varying from 5 to 20. Regarding the energy/time ratio for data transmission, we set $C_{tr} = 10^{-6}$. It is easy to verify that the computation requirement of any task, $C_{k,j}$, is $s-k$ ALU operations. Further, for any task, $C_{k,j}$, the size of data transmitted by any communication task to the task is $s-k$ units in the matrix. We examined two cases with u set to 0.5 and 0.8. In both cases, CCR was selected from $\{1.0, 3.0, 5.0, 8.0, 10.0\}$.

The lifetime improvement achieved by our 3-phase heuristic for the LU factorization algorithm is shown in Figure 3.17. It can be observed that the performance of the heuristic improves when CCR increases or u decreases. The lifetime improvement approaches 8x when $CCR = 10.0$. Also, very few improvement was observed during our simulations by setting CCR beyond 10.0. The least amount of lifetime improvement is around 15% when $u = 0.8$, $CCR = 1.0$, and $s = 20$.

Fast Fourier Transformation (FFT): The recursive, one-dimensional FFT Algorithm is given in Figure 3.18(a). In the figure, A is an array of length l which holds the coefficients



(a) $u = 0.5$ (b) $u = 0.8$

Figure 3.17: Lifetime improvement for the matrix factorization algorithm (10 sensor nodes, 8 voltage levels, 4 channels, 3 modulation levels)

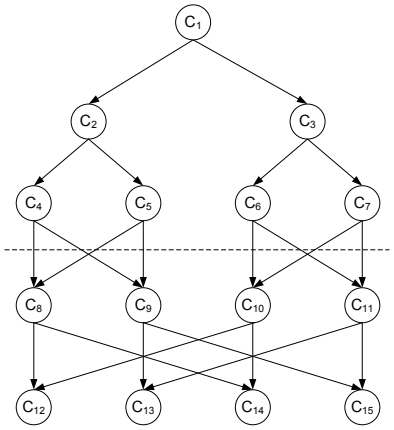
of the polynomial and array Y is the output of the algorithm. The algorithm consists of two parts: recursive calls (lines 3-4) and the butterfly operation (lines 6-7). For an input vector of size l, there are $2 \times l - 1$ recursive call tasks and $l \times \log l$ butterfly operation tasks (we shall be assuming $l = 2^k$ for some integer k). For example, the application graph with four data points is given in Figure 3.18(b) . The 7 tasks above the dashed line are the recursive call tasks, while the 8 tasks below the line are butterfly operation tasks.

We performed simulations used 10 sensor nodes, 8 voltage levels, 4 channels, 3 modulation levels. Regarding the energy/time ratio for data transmission, we set $C_{tr} = 10^{-6}$. The vector size was varied from 4 to 64 incrementing by the power of 2. We also examined two cases with u set to 0.5 and 0.8. In both cases, CCR was selected from $\{1.0, 3.0, 5.0, 8.0\}$.

The lifetime improvement achieved by our 3-phase heuristic for the FFT algorithm is shown in Figure 3.19. Again, the performance of the heuristic improves when CCR increases or u decreases. The lifetime improvement is close to 10x when $CCR = 8.0$ and $l = 64$. The least amount of lifetime improvement is around 75% when $u = 0.8$, $CCR = 1.0$, and $l = 4$.

FFT(A, ω)
1. Set $l = \text{length}(A)$
2. **If** $l = 1$, return A
3. $Y^{(0)} = \text{FFT}((A[0], A[2], \ldots, A[l-2]), \omega^2)$
4. $Y^{(1)} = \text{FFT}((A[1], A[3], \ldots, A[l-1]), \omega^2)$
5. **For** $i = 0$ to $l/2 - 1$ **Do**
6. $\quad Y[i] = Y^{(0)}[i] + \omega^i \times Y^{(1)}[i]$
7. $\quad Y[i + l/2] = Y^{(0)}[i] - \omega^i \times Y^{(1)}[i]$
8. **Return** Y

(a) sequential algorithm



(b) example application graph with 4 points

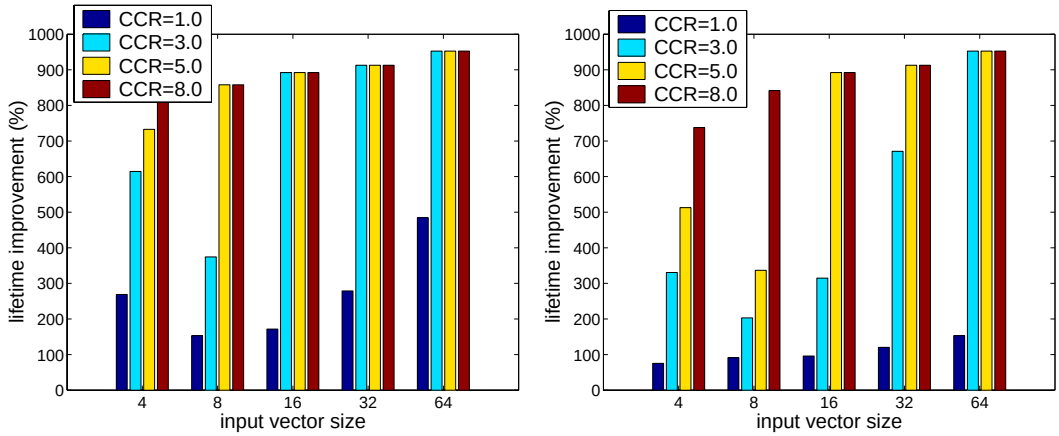
Figure 3.18: Fast Fourier Transformation (FFT) algorithm

Note that the above two example applications have exactly one source task that initially holds the entire data set, implying that data dissemination within the cluster is required. However, our technique is also applicable to applications where data are locally sensed or gathered at each individual sensor node. For example, in Figure 3.18(b), input data can be generated by tasks T4 to T7 through local sensing. Thus, the recursive calls above the dashed line to disseminate the data become unnecessary.

3.7 Concluding Remarks

In this chapter, we have investigated the problem of allocating a real-time application with latency constraint to a single-hop collocated cluster of homogeneous sensor nodes with multiple wireless channels. A new performance metric has been proposed to balance the energy cost among all the sensor nodes by using both voltage scaling and rate adaptation. We have presented both an ILP formulation and a polynomial time heuristic.

We have demonstrated through simulations that for small scale problems with voltage scaling only, a lifetime improvement up to 5x is achieved by the ILP-based approach, compared with the



(a) $u = 0.5$ (b) $u = 0.8$

Figure 3.19: Lifetime improvement for the FFT algorithm (10 sensor nodes, 8 voltage levels, 4 channels, 3 modulation levels)

case where no voltage scaling is used. Also, the performance of the 3-phase heuristic achieves up to 63% of the system lifetime obtained by the ILP-based approach. For large scale problems, a lifetime improvements up to 10x was observed when both voltage and modulation scaling are used. Simulations were also conducted for application graphs from LU factorization and FFT. Our 3-phase heuristic achieves a lifetime improvement of up to 8x for the LU factorization algorithm and an improvement of up to 9x for the FFT algorithm.

Chapter 4

Information Transportation on a Tree Substrate

In this chapter, we study the application of rate adaptation for information routing over a given tree substrate.

4.1 Overview

In last chapter, we have shown the application of voltage scaling and rate adaptation for information processing within a collocated cluster of sensor nodes. In many scenarios, users can access the results of information processing (in general, not necessarily from cluster-based processing) only after they are routed to the base station. Typical communication patterns for such information routing involve multiple source nodes and one sink node. Thus, the corresponding packet flow resembles a reverse-multicast structure, which is often referred to as a *data gathering tree*. It is also well-known that *data aggregation* by each internal node over the tree is crucial for eliminating redundancy among source data and hence reduce the communication load [68].

For applications with small volume data and simple aggregation operation, communication is a significant source of energy dissipation in the process of information routing. Hence, it is

important to design energy-efficient communication strategies. As we pointed out in Chapter 1, rate adaptation is an important technique for improving the energy efficiency of communication. Foreseeing the integration of such a technique into commercial radio modules for sensor nodes, we believe that it is important to analyze the technique in the context of information routing in WSNs.

This chapter studies the problem of scheduling packet transmissions over a data gathering tree using rate adaptation. The main purpose is to explore the energy-latency tradeoffs at the physical layer with respect to the system performance at the application layer. We again consider a real time scenario where the raw data gathered from source nodes must be aggregated and transmitted to the sink within a specified latency constraint. Our technique is applicable to any given aggregation functions. The objective function is to minimize the overall energy dissipation of the sensor nodes in the gathering tree subject to the latency constraint.

4.1.1 Our Contributions

We solve the considered problem using two different but related approaches. In the first approach, we assume a continuously tunable transmission time. A numerical optimization algorithm is developed for solving the off-line version of our problem, where the structure of the data gathering tree and the energy characteristics of all sensor nodes are known *a priori*. While this numerical optimization provides the optimal off-line solution to our problem, it is theoretically difficult to give an upper bound on its running time.

In the second approach, we approximate the continuous transmission time using a set of discrete values. We then derive a recursive presentation of the considered problem, which naturally leads to a dynamic programming based algorithm (DP-Algo) for solving the off-line problem. For a given number of discrete values for the transmission time, DP-Algo solves the off-line problem in polynomial time.

Furthermore, a simple, localized on-line protocol is developed based on discretized transmission time. The key idea is to iteratively identify the sensor node with the highest energy gradient (to be defined later) in the gathering tree and reduce its energy cost when allowed by the latency constraint. In this protocol, each sensor node only needs to perform simple operation based on its local information and the piggybacked information from data messages. The protocol is designed with the aim of self-adaptation to various dynamics in the system, including changes of packet size and latency constraint.

Finally, we evaluate the performance of our algorithms and protocol through extensive simulations. The simulations were conducted for both long and short-range communications. We considered two models of source placement, namely the random source and the event radius source placements [68]. We used the baseline where all sensor nodes transmit the packets at the highest speed (8 bits/symbol) and shut down the radio afterwards. Our simulation results from the scenarios we studied show that compared with this baseline, up to 90% energy savings can be achieved by our off-line algorithms and the on-line protocol. We also investigate the impact of several key network and radio parameters. The adaptability of the protocol is also demonstrated through two run-time scenarios.

4.1.2 Chapter Organization

We briefly discuss the related work in Section 4.2. We describe our underlying network model in Section 4.3. The packet transmission problem is then defined in Section 4.4. Off-line algorithms for the problem are presented in Section 4.5. In Section 4.6, a distributed on-line protocol is described. Simulation results are shown in Section 4.7. Finally, concluding remarks are made in Section 4.8.

4.2 Related Work

Data gathering tree is common to data-centric information routing in WSNs [61, 68]. The construction of the data gathering tree has been studied under various circumstances as previously summarized in Section 1.5.4. For example, several localized tree topology generation mechanisms are compared by Zhou *et. al.* using metrics including node degree, robustness, and latency [137]. When the joint entropy of multiple information sources is modeled as a concave function of the number of sources, a randomized logarithmic approximation algorithm is developed by Goel *et. al.* [47]. By considering a simplified compression model, where the entropy conditioning at nodes only depends on the availability of side information, a hybrid scheme of Shortest Path Tree and Traveling Salesman Path is proved to provide 2-approximation performance for minimizing the overall cost of the data gathering tree [32]. A nice analysis of the impact of spatial correlation on several practical schemes for tree construction [87] indicates that a simple cluster-based routing scheme performs well regardless the correlation among sources. All these works provide the underlying communication substrate above which our algorithms and protocols can be applied for energy minimization.

From wireless communication perspective, rate adaptation have been widely studied to optimize spectral efficiency (e.g., network throughput) subject to the channel conditions in cellular networks [3, 11, 117] or local-area wireless networks [4, 55, 133]. Several recent works [98, 99, 92, 43, 89, 131] have studied the application of rate adaptation for energy conservation, which is closely related to our work.

For a single-hop link, the problem of minimizing the energy cost of transmitting a set of packets subject to a specified latency constraint is studied by Prabhakar *et. al.* [89]. An extension of the problem [43] investigates the packet transmission from one single transmitter to multiple receivers. In both [89] and [43], optimal off-line algorithms and near-optimal on-line solutions are provided. The concept of modulation scaling was first proposed by Schurgers

et. al. [98]. For a single-hop link, policies for adjusting the modulation level are developed for cases where no real-time requirements are imposed [98] or each packet delivery has a deadline to meet [99], Also, modulation adaptation is integrated into multi-packet scheduling with deadline for each packet [99] and the Weighted Fair Queuing (WFQ) scheduling policy [92]. For a multi-hop communication path, modulation scaling is used for balancing the energy cost for all nodes along the path [131].

The real-time latency constraint considered in this paper requires the use of global time-synchronization schemes [38]. Our scenario is similar to the epoch-based data gathering scheme [77], where the length of each epoch actually plays the role of latency constraint. However, prior work has not considered the possibility of using packet-scheduling techniques that trade latency for energy in such a scenario.

To the best of our knowledge, our work is the first to address packet scheduling in a general tree structure. The challenges of our problem are multi-fold. Firstly, the energy functions can vary for different links. It is therefore required to develop general optimization techniques instead of explicit solutions. Secondly, the latency constraint for data gathering in real applications is typically given by considering the aggregation tree as a whole. It is difficult to directly apply the techniques in [43] and [89], as they require explicit latency constraints over each link. Lastly, as described by (2.4) in Section 2.2.2, we consider the non-monotonic energy function of rate adaptation, which is unique to short distance communications in WSNs. This point has not been addressed in previous work. Albeit the above challenges, the tree structure leads us to an extension of the numerical optimization algorithm proposed in [43] as well as a recursive representation of the problem for applying dynamic programming.

4.3 Models and Assumptions

In this section, we first describe the underlying network model, the data gathering tree. We then explain our scheme of computing data aggregation along the tree. For the sake of clarity, we list a summary of notations used in this paper in Table 4.1.

Table 4.1: Table of notations

$T =< V, L >$	the data gathering tree composed by the set of v sensor nodes V and the set of communication links L
$\{V_1, \ldots, V_M\}$	the set of the M leaf nodes in T
V_n	the sink node of T
T_i	the subtree rooted at V_i
Γ	the latency constraint for data gathering over T
s_i	the size of packet transmitted by V_i to its parent
τ_i	the transmission time of s_i
p_i	the path from a leaf node V_i to the sink
Φ_i	the length of p_i, in the metric of transmission time
$w_i(\tau_i)$	the energy function for packet transmission by V_i
m_i	the value of τ_i over $(0, \Gamma]$ that minimizes $w_i(\tau_i)$
$\vec{\tau}$	a schedule of packet transmission, $\vec{\tau} = \{\tau_1, \tau_2, \ldots, \tau_{n-1}\}$
D	the approximation accuracy of DP-Algo
x_i	the latency laxity of V_i
ρ, c	the connectivity and correlation parameters used by our simulation
N	the number of source nodes used by the random source model
S	the sensing range used by the event radius model

4.3.1 Data Gathering Tree

We abstract the underlying structure of the network as a data gathering tree. This is essentially a tree that gathers and aggregates information from multiple sources enroute to a single sink. While there may be transients during the route creation phase, we assume that this tree, once formed, lasts for a reasonable period of time and provides the routing substrate over which aggregation can take place during data gathering.

Since the information flow over the tree is from leaves to the sink, we use a directed graph representation. Let $T = (V, L)$ denote the data gathering tree, where V denotes the set of v sensor nodes, $\{V_i : i \in [v]\}$, and L denotes the set of directed communication links between

the sensor nodes $L = \{L_i : i \in [v-1]\}$. Let M denote the number of leaf nodes in the tree. Without loss of generality, we assume that the sensor nodes are indexed in the topological order with $V_1, \ldots, V_M$ denoting the M leaf nodes and V_v denoting the sink node. For each directed link (i, j), we refer to i as a child of j and j as the parent of i.

Let T_i denote the subtree rooted at any node, V_i, with $T_v = T$. A path in T is defined as a series of alternate nodes and edges from any leaf node, $V_i, i \in \{1, \ldots, M\}$, to V_n, denoted as p_i. We use the notation $V_j \in p_i$ to signify that node V_j is an intermediate node of path p_i.

Raw data is generated by a set of source nodes from V (not necessarily leaf nodes). Data aggregation is performed by all non-sink and non-leaf nodes (referred to as *internal nodes* hereafter). We assume that aggregation at an internal node is performed only after all input information is available at the node – either received from its children, or generated by local sensing if the node is a source node. The aggregated data is then transmitted to the parent node. Let s_i denote the size of the packet transmitted by V_i to its parent. We discuss the computation of data aggregation to determine s_i in the next section.

We assume a simplified communication model with a medium access control (MAC) layer that ensures no collision or interference at a node. Such an assumption can be realized by multi-packet reception (MPR) techniques through frequency or code diversity [119, 116].

For ease of analysis, it is assumed that raw data is available at all source nodes at time 0. Let Γ denote the latency constraint, within which data from all source nodes needs to be aggregated and transmitted to the sink node.

Similar to our assumptions in Chapter 3, we assume that sensor nodes are completely shut down when there is no packet to transmit or receive. We assume that ultra-low power paging or wakeup radios are available for waking up sensor nodes with almost no latency and energy penalties [136, 102]. Also, the computation time and energy costs for generating raw data at source nodes or aggregating data at internal nodes are considered to be negligible.

4.3.2 Data Aggregation Paradigm

Various techniques have been previously proposed for computing aggregates, or joint information entropy, from multiple source nodes. In our study, we adopt the model proposed by Pattern *et. al.* [87] where the joint entropy (or total compressed information) from multiple information sources is modeled as a function of the inter-source distance d, and a pre-specified correlation parameter c, that characterizes the extent of spatial correlation between data. Specifically, let H_1 denote the data size generated from any single source. The compressed information of two sources is calculated as [87]:

$$H_2 = H_1 + \frac{d}{d+c} H_1 \tag{4.1}$$

We assume that the correlation parameter c is the same for any set of sources. Based on (4.1), a recursive calculation of the total compressed information of multiple sources can be developed [87]. We omit the details here.

Although we use the expression in (4.1) as a typical aggregation function, please note that our technique is not limited to this function alone. The only requirement is that we can derive the value of s_i's based on the functions. Thus, even different functions can be used to specify the aggregation at different sensor nodes.

4.4 Problem Definition

A schedule of packet transmission is defined as a vector $\vec{\tau} = \{\tau_i : i = 1, \ldots, v-1\}$, where τ_i is the time duration for packet transmission from sensor node i to its parent. Since a sensor node can transmit its packet only after receiving all input packets from its children, the start time of each transmission is implicitly determined by $\vec{\tau}$. The transmission latency of a path, p_i, is

denoted as Φ_i and calculated as $\Phi_i = \sum_{j:V_j \in p_i} \tau_j$. A schedule is feasible if for any $p_i \in T$, we have $\Phi_i \leq \Gamma$.

While our goal is to improve the energy-efficiency of the system, various objective functions can be developed for interpreting energy-efficiency. For ease of analysis, our objective function is to minimize the overall energy cost for packet transmission of all the sensor nodes in the data gathering tree.

We use the energy model described in Section 2.2.2. Let $w_i(\tau_i)$ denote the energy function of V_i in form of (2.4) with potentially various values of parameters C_{tr}, C_{ele}, and R_s for different links. Let m_i denoting the value of $\tau_i \in (0, \Gamma]$ when $w_i(\cdot)$ is minimized. Note that $w_i(\cdot)$ may vary for different nodes due to variations in packet size and transmission radius (in other words, such information is implicitly embedded into $w_i(\cdot)$). We now formally state the *packet transmission problem* (PTP) as follows:

Given:

a. a data gathering tree T consisting of n sensor nodes,

b. energy functions for each link $(i, j) \in E$, $w_i(\tau_i)$, and

c. the latency constraint, Γ;

find *a schedule of packet transmission, $\vec{\tau} = \{\tau_i : i \in [n-1]\}$, so as to minimize*

$$f(\vec{\tau}) = \sum_{i=1}^{n-1} w_i(\tau_i) \tag{4.2}$$

subject to

$$\forall p_i \text{ in } T, \Phi_i = \sum_{j:V_j \in p_i} \tau_j \leq \Gamma \ . \tag{4.3}$$

The above formulation differs from the problem defined in [43] in two key aspects. (1) We employ a tree structure packet flow where the latency constraint is imposed on each path of

the tree. (2) The non-monotonic energy model in Section 2.2.2 indicates the presence of an upper-bound on the transmission time of each packet, i.e., to optimize PTP, we should have $\tau_i \leq m_i$, for each $i \in [n-1]$. The consequences of such differences are discussed in Section 4.5.1.

We consider two versions of PTP, an off-line version and an on-line version. In the off-line version, the structure of the data gathering tree and the energy functions for all sensor nodes are known a priori. Centralized algorithms can be developed for solving the off-line version. In the on-line version, each sensor node only has local knowledge about its own radio status and can communication with its parent and children. Hence, distributed on-line protocols are needed to locally adapt the transmission time of each sensor node to achieve global energy minimization.

4.5 Off-line Algorithms for PTP

In this section, we consider an off-line version of PTP (called OPTP) by assuming that the structure of the aggregation tree and the energy functions for all sensor nodes are known a priori. We first describe an extension of the MoveRight algorithm [43] to get optimal solutions for OPTP. A faster dynamic programming based approximation algorithm is then presented.

4.5.1 A Numerical Optimization Algorithm

Since we must have $\tau_i \leq m_i$ in an optimal solution to OPTP, the latency of a path does not necessarily equal Γ. Moreover, let V_i denote an internal node. For any optimal solution to OPTP, we show that the first derivative of the energy function of V_i equals the sum of the first derivatives of the energy functions of all children of V_i.

Lemma 1 *A schedule, $\vec{\tau^*}$, is optimal for OPTP iff*

1. *for any node V_i with $\tau_i^* < m_i$, the length of at least one path that contains V_i is equal to Γ; and*

2. *for any internal node, V_i, we have*

$$\dot{w}_i(\tau_i^*) = \sum_{(j,i)\in E} \dot{w}_j(\tau_j^*) \ . \tag{4.4}$$

The proof of the lemma is presented in Appendix A.

The problem proposed in [43] is to schedule multiple packet transmission over a single transmitter multiple receiver connection, where the ready time of packets can differ from each other. A special case of the problem is to assume all packets are of equal size and ready at time 0. This special case can also be regarded as a special case of the OPTP problem where (1) the aggregation tree degenerates into a pipeline of sensor nodes – the latency constraint is imposed over exactly one path; and (2) all energy functions are monotonically decreasing. The MoveRight algorithm proposed in [43] can be directly applied to solve such a special case.

Begin

1. Set $k \leftarrow 0$ // initialize iteration counter
2. **For** $(i, v) \in E$, set $\tau_i^k \leftarrow \min\{\Gamma, m_i\}$ // initialize transmission time for links to the sink
3. **For** $(i, j) \in E$ such that $j \neq n$, set $\tau_i^k \leftarrow 0$ // initialize transmission time for other links
4. Set $flag \leftarrow 0$ // flag to keep track of convergence in the iterations
5. **While** $flag = 0$
6. $\quad k \leftarrow k + 1$ // increment the iteration counter by 1
7. $\quad$ **For** each V_i with i from $v - 1$ downto M+1 // perform local optimization for each internal node
8. $\quad\quad (\{\tau_j^k\}_{(j,i)\in E}, \tau_i^k) \leftarrow \text{best}(\{\tau_j^{k-1}\}, \tau_i^{k-1})$ // move right the start time of transmission from V_i
9. $\quad$ **For** $(i, v) \in E$
10. $\quad\quad$ Set $\tau_i^k \leftarrow \min\{m_i, \Gamma - (\max_{V_i \in p_j}\{L_j\} - \tau_i^k)\}$ //increase the transmission time for links to the sink
11. if $\vec{\tau}^k = \vec{\tau}^{k-1}$, $flag \leftarrow 1$ // check convergence

End

Figure 4.1: Pseudo code for EMR-Algo

We now extend the MoveRight algorithm to solve OPTP in a general-structured aggregation tree with non-monotonic energy functions. The pseudo code for the extended MoveRight algorithm (EMR-Algo) is shown in Figure 4.1. In the figure, τ_i^k denotes the value of τ_i in the k-th iteration. Initially, we set the starting time for all packet transmission to zero – the transmission time for all the links to the sink is set to $\min\{\Gamma, m_i\}$, while the transmission time for the rest links is set to 0 (Steps 2 and 3). The main idea is to iteratively increase (move right) the starting times of packet transmissions, so that each move locally optimizes our objective function. Finally, this iterative local optimization leads to a globally optimal solution.

The best$(\cdot)$ function returns the transmission time for node V_i and its children so that Lemma 1 holds for the subtree formed by V_i and its parent and children, with respect to the invariant that $\tau_j^k \leq m_j$ for any node V_j in the subtree. When best$(\cdot)$ is called upon the subtree around V_i, the transmission for all the links not within the subtree remain fixed, i.e., the starting time of transmissions from the children of V_i and the ending time of the transmission from V_i are fixed. We prove in Theorem 1 that the starting time of the transmission from V_i will never be decreased by calling best$(\cdot)$. Hence, in best$(\cdot)$, the locally optimal starting time of the transmission from V_i is obtained by a binary search between the original starting time and the ending time of the transmission. Step 10 is important as it moves right the complete time of transmissions on links to the sink. This movement stops when the latency constraint is reached.

The proposed EMR-Algo is distinguished from the MoveRight algorithm in two key respects (recall the difference discussed in Section 4.4 between our problem and the one defined in [43]). (1) The best$(\cdot)$ function respects Lemma 1 regarding the optimality in a tree structure. (2) The transmission time for any $V_i \in V$ is bounded by m_i, enforced by lines 2, 8 and 10.

The correctness of EMR-Algo can be proved by exploring the convexity property of the energy functions. Let $\vec{\tau^*} = \{\tau_1^*, \ldots, \tau_{v-1}^*\}$ be the optimal schedule. Let $\theta_i^* = 0$, for $i =$

$1, \ldots, M$; and $\theta_i^* = \max_{(j,i)\in E}(\theta_j^* + \tau_j^*)$, for $i = M, \ldots, v-1$. As previously stated, $\{\tau_i^k : k = 1, \ldots, v-1\}$ indicate the transmission time of nodes $V_1, \ldots, V_{v-1}$ after the k-th pass of EMR-Algo. Let $\theta_i^k = 0$, for $i = 1, \ldots, M$, and $\theta_i^k = \max_{(j,i)\in E}(\theta_j^k + \tau_j^k)$, for $i = M, \ldots, v-1$.

Theorem 1 *Let θ_i^k and θ_i^*, $i = 1, \ldots, n-1$ be as defined before. Then*

1. $\theta_i^k \leq \theta_i^{k+1}$;
2. $\theta_i^k \leq \theta_i^*$; *and*
3. $\theta_i^\infty = \theta_i^*$.

The proof of Theorem 1 is developed based on the proof of Theorem 1 in [43] and is detailed in Appendix A.

The convergence speed of EMR-Algo depends on the structure of the aggregation tree and the exact form of the energy functions. It is therefore difficult to give a theoretical bound on the number of iterations. In Section 4.7, we show running time of EMR-Algo for simulated problems. However, by approximating $w_i(\tau)$ with a set of interpolated discrete values, we develop a much faster approximation algorithm based on dynamic programming. We present the approximation algorithm in Section 4.5.2.

4.5.2 A Dynamic Programming Based Approximation Algorithm

For ease of analysis, we assume that for each sensor node, D discrete values are evenly distributed over $[0, \Gamma]$ in the domain of τ. Let ε be the difference between two adjacent values. That is $\varepsilon = \frac{\Gamma}{D}$. Hereafter, D is called the approximation accuracy. A higher value of D leads to a more accurate approximation of the energy function. By changing D, we can explore the tradeoffs between the quality of the solution and the time cost of the algorithm.

Let $g(V_i, t)$ denote the minimal overall energy dissipation of a subtree rooted at V_i within latency constraint t. The original OPTP problem can be expressed as $g(V_v, \Gamma)$. It is clear that

for any sensor node V_i, $g(V_i, t)$ can be computed as the sum of (a) the energy dissipation for packet transmission by the children of V_i, and (b) the energy dissipated by transmitting packets within the subtrees rooted at each child of V_i. Additionally, the packet transmission time from any child of V_i can take $\frac{t}{\varepsilon}$ values, namely $\varepsilon, 2\varepsilon, \ldots, t$. Therefore, we have the following recursive representation of $g(V_i, t)$:

$$g(V_i, t) = \begin{cases} w_i(t), \text{ for } 1 \leq i \leq M & \text{(4.5a)} \\ \sum_{(k,i) \in E} (\min_{j=1}^{\frac{t}{\varepsilon}} \{w_k(j\varepsilon) + g(V_k, t - j\varepsilon)\}), \text{ otherwise} & \text{(4.5b)} \end{cases}$$

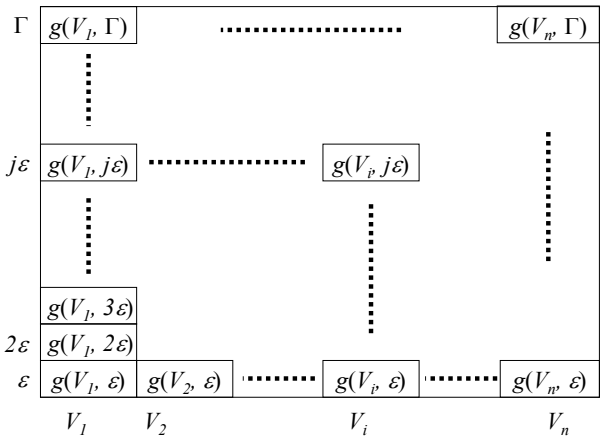


Figure 4.2: The $g(\cdot)$ table computed by DP-Algo

The above representation is suitable for a dynamic programming based algorithm (DP-Algo for short). DP-Algo can be viewed as a procedure to build a table of size $D \times v$ (Figure 4.2). The i-th column from the left side corresponds to sensor node V_i, while the j-th row from bottom-up corresponds to $j\varepsilon$. After the execution of DP-Algo, the cell crossed by the j-th row and the i-th column shall contain the value of $g(V_i, j\varepsilon)$.

To build the table, we start from the bottom left cell that contains $g(V_1, \varepsilon) = w_1(\varepsilon)$. The table is then completed column by column, from left to right. To calculate the value of $g(V_i, j\varepsilon)$

for $i > M$, we need to compare, for each child of V_i, j different values by varying the packet transmission time of the child. Therefore, the time cost for building up the table is $O(D^2(v+l))$, which is polynomial with respect to v and l for a fixed D.

A Special Case for Modulation Scaling:In practice, the modulation levels are typically set to positive even integers. Based on equation 2.3, it can be verified that the τ_i's resulted from different modulation levels are not evenly distributed among $[0, \Gamma]$. Thus, DP-Algo cannot be directly applied. However, one practical method is to, for each i, set τ_i obtained by EMR-Algo or DP-Algo to the largest time duration smaller than τ_i that can be achieved by an available modulation level. We call the above method the *rounding procedure*. Such a rounding procedure may affect the performance of DP-Algo. As shown in Section 4.7, the performance degradation is around 10% for loose latency constraints and 50% for stringent latency constraints.

Another issue with modulation scaling is that the maximal or minimal transmission time can be bounded, due to the lowest and highest settings for the modulation level. Such boundaries can be captured by filling the corresponding cells in the table with infinity.

4.6 Distributed On-line Protocol

The algorithms presented in Section 4.5 all assume a complete knowledge of the data gathering tree. However, the discrete approximation of the energy function motivates a simple on-line distributed protocol that relies on local information of sensor nodes only. The key idea of the protocol is to identify the sensor nodes with the largest energy gradient on each path of the tree. Here, energy gradient is defined as the amount of energy that can be saved by increasing the transmission time of the sensor node by ε. We then increase their transmission time if the latency constraint will not be violated. We repeat the above procedure until either the latency constraint is reached for all paths or the energy cost of the gathering tree is minimized.

To facilitate the on-line scheduling, we make the following assumptions:

1. Some local unique neighbor identification mechanisms are available at each sensor node for identifying the parent and children.

2. Every node V_i can derive the time cost for data gathering within the subtree rooted V_i.

3. Every sensor node can measure its current power consumption, and hence its energy gradient.

4. Interference among sensor nodes is handled by MPR techniques.

The local identifier in assumption 1 is commonly implemented in protocols such as Directed Diffusion [61]. Assumption 2 can be fulfilled by attaching a time stamp to each packet from the leaf nodes (we shall be assuming that time synchronization schemes, such as [38], are available). In assumption 3, the power consumption and energy gradient of a sensor node can be determined using the system parameters provided by the hardware vendors and the operating configuration of the system, such as the modulation level. Assumption 4 can be satisfied by intentionally setting the latency constraint to be tighter than the actual constraint for accommodating the incurred time cost for resolving collisions.

Moreover, we define the *latency laxity* of a node as the maximal amount of time that can be used to increase the transmission time of the node without violating the latency constraint. Let x_i denote the latency laxity of V_i. The latency laxity of each node is dynamically maintained during the protocol to verify if the transmission time of the node can be safely increased.

In the following, we first describe the local data structure maintained at each sensor node. A distributed adaptation policy for minimizing the energy cost is then presented.

Local Data Structure: Each sensor node, V_i, maintains a simple local data structure (r, τ_i, τ_d). The flag r equals one if V_i is the node with the highest *positive* energy gradient in subtree T_i, and zero otherwise. Field τ_i is the time cost for transmitting the packet from V_i to its parent, while τ_d records the time cost of the longest path, *excluding* τ_i, in T_i.

The local data structure is maintained as follows. Every leaf node piggybacks its energy gradient to the outgoing packet. Once a sensor node, V_i, receives packets from all its children, the node compares the energy gradients piggybacked to each packet and the energy gradient of its own. The value of r at V_i is then set accordingly. If V_i is not the sink, the largest energy gradient from the above comparison is piggybacked to the packet sent to the parent of V_i. The above procedure continues till all the sensor nodes have the correct value of r. Fields τ_i and τ_d can be easily maintained based on the above assumptions.

Adaptation Policy: The sink node periodically disseminates a feedback packet to its children that contains the value of its local τ_d and the difference between Γ and τ_d, denoted as δ. Basically, δ is the latency laxity of nodes on the longest path of the data gathering tree.

Once a sensor node V_i receives the feedback packet from its parent, it performs the following adaptation. To distinguish from the field τ_d in V_i's local data, let τ'_d denote the field τ_d in the feedback packet. First, the latency laxity of V_i can be calculated as $x_i = \delta + \tau'_d - (\tau_i + \tau_d)$. This is because $\tau_i + \tau_d$ is the time cost of T_i; τ'_d is the time cost of the longest path in the subtree rooted at V_i's parent (excluding the transmission time of V_i's parent); and δ is actually the latency laxity of nodes on this longest path. Then, V_i takes one of the following actions.

1. If $\delta < 0$, the transmission time for packet from V_i is decreased by a factor of β, where β is a user-specified parameter. The feedback packet is then forwarded to all of V_i's children.

2. If $r = 1$ and $x_i \geq \varepsilon$, the transmission time of V_i is increased by ε. The local data structure at V_i is updated accordingly; and the feedback packet is suppressed.

3. Otherwise, the feedback packet is updated by setting $\delta = x_i$ and $\tau'_d = \tau_d$. The updated packet is then forwarded to all children of V_i.

The rationale behind the above adaptation policy is that when the latency constraint is violated, all the sensor nodes send out packets with an increased rate (action 1). If V_i is

the node with the largest positive energy gradient in T_i and the latency laxity allows, the transmission time of V_i is increased (action 2). Otherwise, the latency laxity of V_i is recorded in the feedback packet and the sensor nodes in T_i are recursively examined (action 3).

Discussion: During each dissemination of the feedback packet, the proposed on-line protocol increases the transmission time for at most one sensor node per path. Such an increment is guaranteed not to violate the latency constraint. Therefore, the on-line protocol converges after the latency constraint is reached by all paths, or $\tau_i = m_i$, for each $V_i \in V$. We assume that each sensor node has q discretized transmission time. Before the protocol converges, a feedback packet would increase the transmission time for at least one sensor node when it traverses the data gathering tree. Thus, the protocol converges after the dissemination of at most nq feedback packets, where n is the number of sensor nodes in the tree.

Various tradeoffs can be explored in implementing the above protocol. Ideally, the adaptation should be performed under a stable system state. Thus, the period α for disseminating the feedback packet should be large enough to accommodate oscillation in system performance. However, a larger period means a longer convergence process with greater energy cost. There is also a tradeoff involved in selecting the value of β. A larger value of β leads to higher transmission speed when the latency constraint is violated. However, extra energy cost is caused if the violation is not dramatic. Intuitively, β should be related to the severity of the violation, which is indicated by the value of δ.

Another option to handle latency violations is to repeatedly reduce the transmission time of the sensor nodes with the smallest energy gradient till the latency constraint is satisfied. Compared with the proposed technique that simultaneously reduces the transmission time of all sensor nodes, such an option is more aggressive in the sense of reducing the incurred increment in energy cost. However, it requires more sophisticate control protocol and more importantly, increases the response time in handling latency violations.

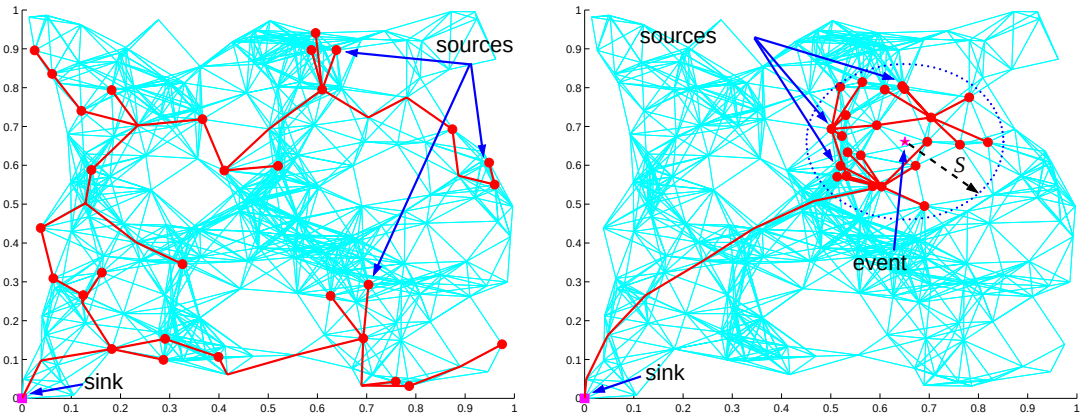
The above protocol actually does not require the discretized transmission time to be evenly distributed with distance ε. In the example of modulation scaling, the set of transmission times are generated based on $\frac{s}{b \cdot R}$ by varying b within $[2, 4, 6, ...]$. The distance between adjacent transmission times decreases with b. This can be handled by the following slight modification to the protocol. First, in action 1, after decreased by a factor of β, the transmission time is lower-rounded to the closest transmission duration . Second, in action 2, the latency increment is determined by the current value of b, instead of being a fixed ε. We will use this modified protocol for our on-line simulation.

4.7 Simulation Results

To conduct the simulations, a simulator was developed using the PARSEC [86] software, which is a discrete-event simulation language. The purposes of the simulations are (1) to demonstrate the energy gain achieved by our algorithms compared with the baseline; (2) to evaluate the impact of several key system parameters to the performance of our algorithms; and (3) to show the energy saving and the adaptation capability of our on-line protocol in various run-time scenarios.

4.7.1 Simulation Setup

The transmission speed of sensor nodes is continuously tunable by setting modulation level within $(0, \infty)$, except for the special case of modulation scaling where modulation level can only be integer even values within $[2, 8]$. Hence, when modulation scaling is used, the highest data rate is 8 Mbps and the lowest data rate is 2 Mbps. The baseline in our simulations is to transmit all packets at the highest speed (i.e., 8 Mbps), and shut down the radios afterward. This policy is use, for example, in the PAMAS protocol [91] and the DMAC protocol [74].



(a) Random sources model (number of sources $N = 30$)

(b) Event radius model (sensing range $S = 0.2$)

Figure 4.3: Two example data gathering trees generated by the random sources and event radius models, respectively (connectivity parameter $\rho = 0.15$)

A sensor network was generated by randomly scattering 200 sensors in a unit square. The sink node was put at the left-bottom corner of the square. The neighbors that a sensor node can directly communicate is determined by a connectivity parameter, $\rho \in (0, 1]$. Specifically, two sensor nodes can communicate with each other only if the distance between them is within ρ. We used two models for generating the location of the data sources, namely the random sources (RS) model and the event radius (ER) model. In the RS model, N (the number of sources) out of 200 sensor nodes are randomly selected to be the sources, whereas in the ER model, all sources are located within a distance S (essentially the sensing range) of a randomly chosen "event" location. For both models, the Greedy Incremental Tree (GIT) algorithm [68] was used for constructing the data gathering tree. In Figure 4.3, we illustrate two example data gathering trees generated based on the RS and ER models, respectively.

The energy function used in the simulation was in the form of (2.4). Unless otherwise stated, we set $R_s = 10^6$ and $C_{ele} = 10^{-8}$ for all the sensor nodes, while the value of C of a sensor node was determined by the distance from the node to its parent in the tree. Specifically, we assume

a d^2 power loss model, where d is the distance between a node and its parent. Then, for node V_i, we have $C_i = C_{base} \cdot (\frac{d}{\rho})^2$. Based on our analysis in Section 2.2.2, C_{base} was set to 6×10^{-9} for the long-range communication and 3×10^{-10} for the short-range communication.

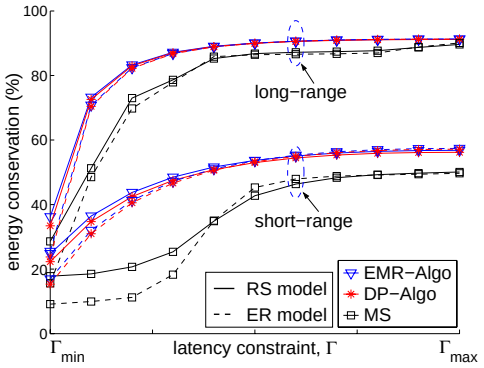
During our simulation, the latency constraints Γ was determined as follows. We define the shortest time cost, Γ_{min} of a gathering tree as the transmission latency of the longest path in the tree when all sensor nodes transmit at the highest speed (8 Mbps). On the other hand, the longest time cost, Γ_{max} of the gathering tree is defined as the transmission latency of the longest path in the tree when every sensor node V_i sends its packet using time $\min\{m_i, \frac{s_i}{2}\}$. In the above definition, the term m_i comes from the fact that it is not energy beneficial for V_i to transmit its packet using time beyond m_i; the term $\frac{s_i}{2}$ is due to the lower bound of modulation level in our simulation. Therefore, Γ was adjusted between Γ_{min} and Γ_{max}.

4.7.2 Performance of the Off-Line Algorithms

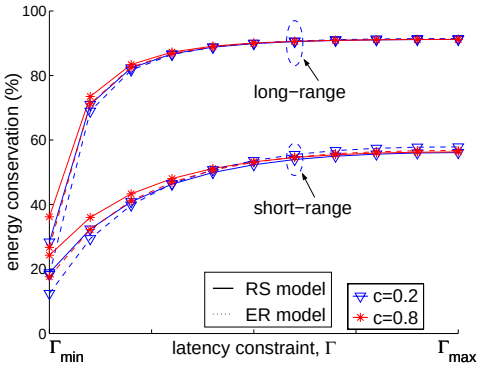
The performance metric is defined as the percentage of energy savings achieved by using our techniques, compared with the baseline. In the simulation, the approximation accuracy for DP-Algo, D was set to 100. The size of raw data generated by source nodes was set to 200 bits.

In Figure 4.4, we show the energy saving achieved by our off-line algorithms for both the RS and ER models. For both models, we show the results for long and short range communication. Each data point in the figures is averaged over more than 100 instances such that it has a 95% confidence interval with a 5% (or better) precision. In the following, we focus on analyzing the results for the RS model; similar analysis can be made for results of the ER model.

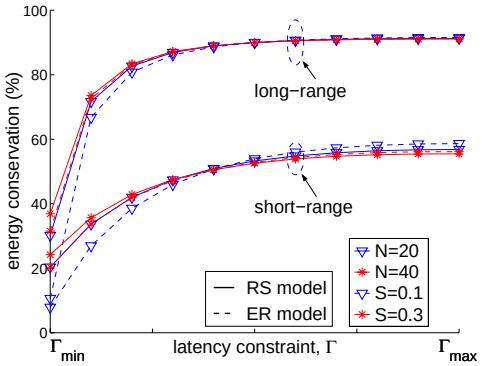
Performance Overview: In Figure 4.4(a), we investigate the performance of algorithms including EMR-Algo, DP-Algo and the special case of DP-Algo for modulation scaling (denoted as MS in the figure) when varying Γ from Γ_{min} to Γ_{max}. The first thing to notice is that



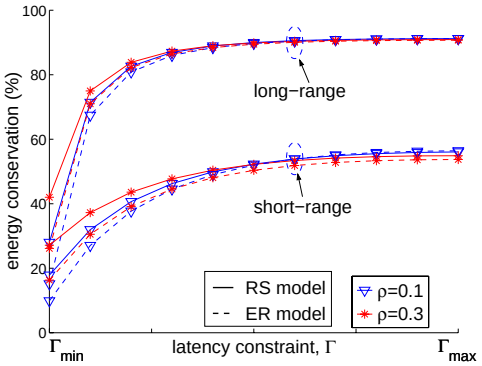
(a) Performance overview ($c = 0.5$, $\rho = 0.15$, $N = 30$, $S = 0.2$,)



(b) Impact of the correlation parameter c ($\rho = 0.15$, $N = 30$, $S = 0.2$)



(c) Impact of the number of sources N or sensing range S ($c = 0.5$, $\rho = 0.15$,)



(d) Impact of the connectivity parameter ρ ($c = 0.5$, $N = 30$, $S = 0.2$)

Figure 4.4: Performance of our off-line algorithms (c: correlation parameter, ρ: connectivity parameter, N: number of sources, S: sensing range)

when Γ approaches Γ_{max}, our algorithms achieve more than 90% energy saving for the long-range communication, and around 50% for the short-range communication. Even when $\Gamma = \Gamma_{min}$, EMR-Algo and DP-Algo can still save more than 30% of the energy for long-range communication and 20% for short-range communication.

The reason for successful energy saving even when $\Gamma = \Gamma_{min}$ is two-fold. First, modulation level in EMR-Algo is allowed to be varied between $(0, \infty)$ instead of $[2, 8]$. Second, Γ equals the transmission time of the longest path in the data gathering tree. Thus, energy can still be

Table 4.2: The miss rate of MS (based on simulated instances for Figure 4.4(a))

source model	communication scenario	# of successful instances	# of failed instances	total number of instances	miss rate (%)
RS	long-distance	102	28	130	21
	short-distance	104	16	120	13
ER	long-distance	324	66	390	17
	short-distance	352	38	390	10

reduced for nodes not on the longest path. On one hand, when there exists only one path in the tree, no energy can be saved when $\Gamma = \Gamma_{min}$. On the other hand, when the tree forms a star-like structure, all links, except the longest ones, can be optimized for energy savings when $\Gamma = \Gamma_{min}$. This also explains the performance degradation of our algorithms in the ER model, compared with the performance in the RS model. Specifically, as illustrated by Figure 4.3, the gathering tree for the ER model forms a small cluster connected to the sink by a linear array of sensor nodes, while the tree for the RS model is more close to a star-like structure.

The plot shows that the performance of DP-Algo is quite close to the performance of EMR-Algo. However, the performance of MS quickly degrades when Γ decreases. From Figure 2.3, the first derivative of the energy function decreases fast as τ tends to 0. Thus, when Γ is small, the rounding procedure for solutions with high modulation levels leads to large performance loss.

During our simulation, we also observed that when $\Gamma = \Gamma_{min}$, MS fails to find feasible solutions for some problem instances, due to the rounding procedure of the approximated solutions from DP-Algo. The ratio of instances that MS fails over the total number of instances is between 10 - 20% in our simulations. We define the miss rate of MS as the ratio of number of instances that MS fails to find a feasible solution over the number of total problem instances. The miss rate for the performed simulations is given in Table 4.2.

The simulation was performed on a SUN Blade1000 with a 750 MHz SUN UltraSPARC III processor. The running time of EMR-Algo is between 0.5 to 3 second, whereas the running time of DP-Algo is around 0.01 second.

Impact of Network Parameters: Figure 4.4(b) shows the energy conservation achieved by DP-Algo with respect to variations in c and Γ. It was observed that for a fixed Γ, the energy gain of DP-Algo slightly increases when c increases. This is because a smaller value of c causes a larger size of data packet after aggregation. Thus, the energy cost by links close to the sink node dominates the overall energy cost of the tree. It is however difficult to reduce the energy cost of these links since they have high a likelihood of lying on the longest path of the tree.

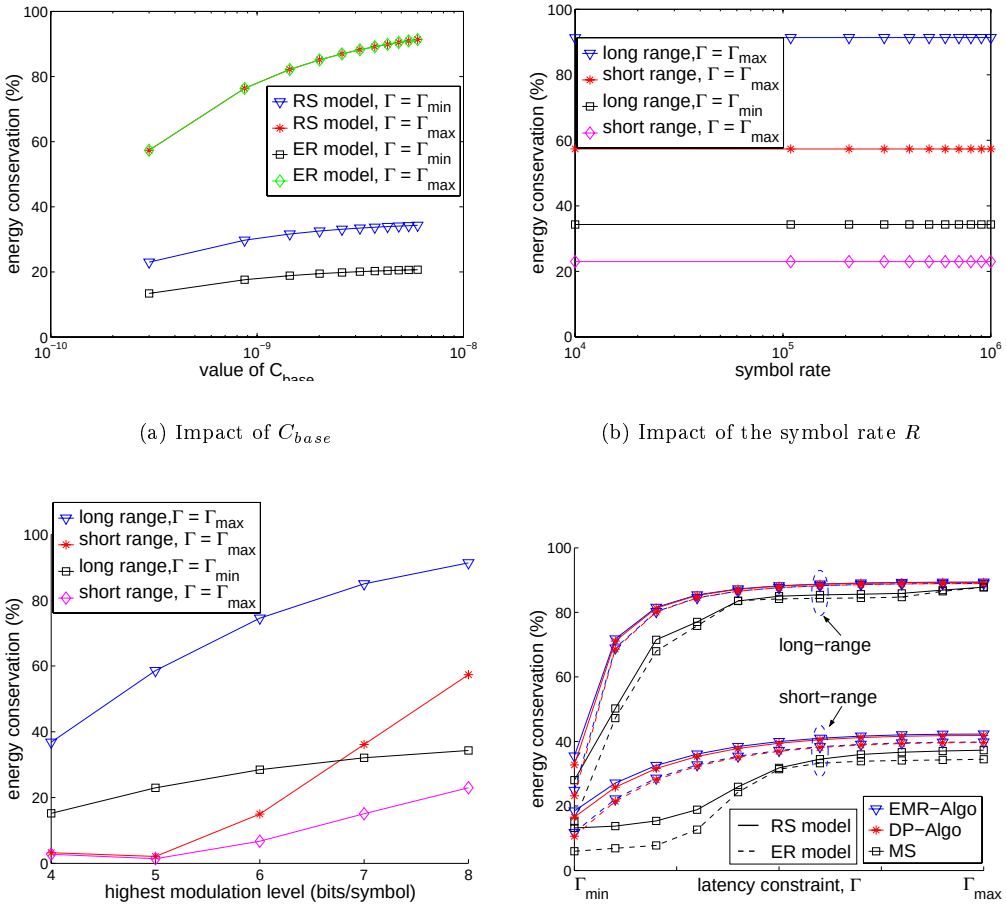
Figure 4.4(c) plots the performance of DP-Algo with respect to variations in N and Γ. It can be seen that when Γ is close to Γ_{min}, the energy gain of DP-Algo increases as the number of sources increases. This is because a larger number of sources offers more opportunities for the optimization of links on paths other than the longest one.

Figure 4.4(d) demonstrates the performance of DP-Algo with respect to variations in ρ and Γ. It can be observed that the energy saving of DP-Algo increases when ρ increase. This is understandable since a large ρ reduces the height of the data gathering tree (the extreme case is a star-like tree formed by setting $\rho = 1$).

Together, the above results suggest that DP-Algo is quite robust with respect to variations in different system parameters, including c, ρ, N, and S.

Impact of the radio parameters: In Figure 4.5(a), we show the impact of radio parameter C_{base} on the performance of DP-Algo. In the figure, the x-axis represents the value of C_{base} from 3×10^{-10} to 6×10^{-9} in logarithmic scale. As expected, the energy conservation achieved by DP-Algo increases with C_{base}. Also, when $\Gamma = \Gamma_{max}$, there is almost no difference in the performance of DP-Algo under either RS or ER models; whereas when $\Gamma = \Gamma_{min}$, a performance degradation of 9-14% is observed for the ER model compared with the RS model.

To evaluate the impact of symbol rate R_s, we varied R_s from 10 KBaud to 1 MBaud. Consider the modulation level within $[2, 8]$, the above range of R_s reflects a bit rate of 20 to 80 Kbps when $R_s = 10$ KBaud and 2 to 8 Mbps when $R_s = 1$ MBaud. We show the performance



(a) Impact of C_{base}

(b) Impact of the symbol rate R

(c) Impact of the highest modulation level

(d) Impact of the start-up energy (1 μJ)

Figure 4.5: Impact of radio parameters (correlation parameter $c = 0.5$, connectivity parameter $\rho = 0.15$, number of sources $N = 30$, sensing range $S = 0.2$)

of DP-Algo for RS model in Figure 4.5(b). It can be seen that the energy saving is almost the same throughout the variation of R. This is quite understandable, since from (2.4), the performance ratio of DP-Algo to the baseline is determined by b, but not R_s.

We further investigate the energy saving of DP-Algo under different settings of the highest modulation level of the radio. In Figure 4.5(c), we show the energy saving achieved by DP-Algo in RS model when the highest modulation level is varied from 4 to 8. As expected, lower highest

modulation level results in less energy saving. When the modulation level is restricted within [2, 4], less than 20% energy saving is achieved for both long and short range communication.

We now show the impact of start-up energy of radios, which was estimated as 1 μJ [93]. In each epoch, the radio of each sensor node is started exactly once. Figure 4.5(d) shows the performance of our algorithms with start-up energy. Though the impact of the start-up energy to the long-range communication is almost negligible, we observe a decrease of 6-15% in energy conservation for the short-range communication (compared with Figure 4.4(a)). This is because the start-up energy is comparable to the transmission energy for short-range communication. However, the energy conservation of our algorithms is still considerable in the studied scenario.

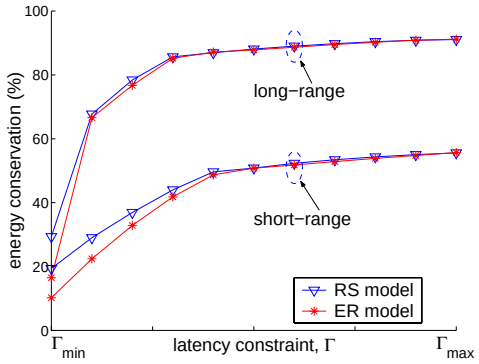
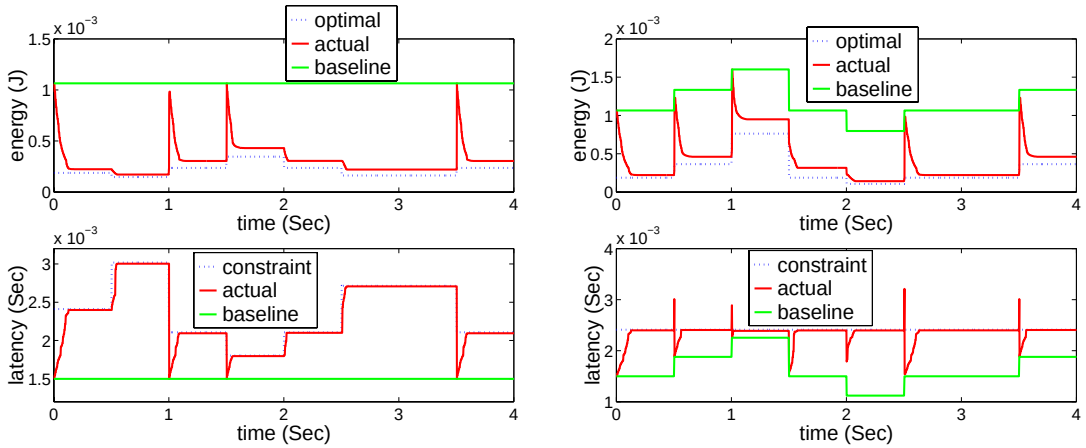


Figure 4.6: Performance of the on-line protocol (correlation parameter $c = 0.5$, connectivity parameter $\rho = 0.15$, number of sources $N = 30$, sensing range $S = 0.2$)

4.7.3 Performance of the On-Line Protocol

Energy Conservation: We show the energy conservation achieved by the on-line protocol in Figure 4.6. The simulated on-online protocol is based on modulation scaling where available modulation levels are even integers within $[2, 8]$. The presented data is averaged over more than 150 problem instances and has a 95% confidence interval with a 10% (or better) precision. In each instance, we generated a sensor network with 200 randomly dispersed sensor nodes. After randomly selecting 20 source nodes, the data gathering tree was then generated using GIT.

When the latency constraint approaches Γ_{min}, there is slight performance degradation compared with DP-Algo from Figure 4.4(a). Specifically, for the RS model, we observe around 4% less energy conservation for long-range communication and 3% for short-range communication. This is reasonable considering the fact that only 4 options are available to set the transmission time for each sensor in the on-line protocol, instead of the fine granularity adjustment of the transmission time in DP-Algo. Moreover, the on-line protocol actually outperforms the modulation scaling case (MS) shown in Figure 4.4(a), implying a large performance degradation of the rounding technique used by MS.



(a) Scenario A

(b) Scenario B

Figure 4.7: Adaptability of the on-line protocol (correlation parameter $c = 0.5$, connectivity parameter $\rho = 0.15$)

Adaptability to System Variations: Our simulations were performed based on the tree shown in Figures 4.3(a) that has 44 sensor nodes out of which 30 are source nodes. Again, we assume that modulation scaling is used by all the nodes with the available modulation levels being even numbers within $[2, 8]$. The data gathering was requested every 2 milliseconds. For the sake of illustration, we set $\alpha = 4$ milliseconds, and $\beta = 10$. Two run-time scenarios, namely A and B, were investigated to demonstrate the efficiency and adaptability of our protocol.

Scenario A: We fixed s at 200 bits. It can be calculated that $\Gamma_{min} \approx 1.5$ milliseconds and $\Gamma_{max} \approx 4.5$ milliseconds. We set Γ to 2.4, 3, 2.1, 1.8, 2.1, 2.7, 2.1 milliseconds at 0, 0.5, 1, 1.5, 2, 2.5, and 3.5 seconds, respectively. In real life, such variations can be caused by for example, change of user requests.

We depict the energy cost and latency for data gathering over 4 seconds in Figure 4.7(a), where the optimal solutions are obtained by using EMR-Algo. It can be observed that when Γ is fixed, the actual energy cost gradually decreases till it is close to the optimal, while the latency approaches the constraint. At time 1 second, Γ is varied from 3 milliseconds to 2.1 milliseconds, which causes a violation of the latency constraint. Due to the feedback mechanism, the transmission latency dramatically decreases as the modulation settings of all the sensor nodes are restored to higher levels. Consequently, the energy cost is also increased. After that, the energy cost drops again as time advances. Moreover, by setting $\beta = 10$, the modulation levels of the sensor nodes were restored to the highest levels when a violation is detected, reflected by the high peaks in the energy curve.

Scenario B: We set $\Gamma = 2.1$ milliseconds, while setting s to 200, 250, 300, 200, 150, 200, and 250 at 0, 0.5, 1, 1.5, 2, 2.5, and 3.5 seconds, respectively. In real life, the change of packet size may be caused by variations in gathered information, or the correlation parameter at sensor nodes. The results is illustrated in Figure 4.7(b), where the optimal solutions are also obtained by using EMR-Algo. An analysis similar to the one in scenario A can be performed.

In short, our on-line protocol is capable of saving significant energy in the studied scenarios. The ability of the protocol to adapt the packet transmission time with respect to the changing system parameters is also demonstrated.

4.8 Concluding Remarks

In this chapter, we have studied the problem of scheduling packet transmissions over a data aggregation tree by exploring the energy-latency tradeoffs. For the off-line version of the problem, we have provided (a) a numerical algorithm for optimal solutions, and (b) a faster approximation algorithm based on dynamic programming. Our simulation results show that between 20% up to 90% energy saving can be achieved by the algorithms. We have investigated the performance of our algorithms with different settings of several key system parameters. Furthermore, we have proposed a distributed on-line protocol that relies on local information only. Our simulation results show that the energy saving achieved by the protocol is between 15% up to 90%. Also, the ability of the protocol to adapt the packet transmission time upon variations in the system parameters were demonstrated through several run-time scenarios.

Chapter 5

Information Routing with Tunable Compression

In this chapter, we investigate the construction of a data gathering tree for joint information routing and tunable compression.

5.1 Overview

In last chapter, we assume that maximal compression is used for data aggregation. In other words, the data is compressed as much as possible for reducing communication load. This method is reasonable for applications with small amount of data volume and simple compression operations (e.g., temperature sensing), indicating that communication cost dominates the computation cost. However, in case of more advanced and computation-intensive applications with heavy data flow (including streaming media, video surveillance, and image-based tracking), compression of complex data set is envisioned to have comparable amount of energy cost as that of wireless communication. A similar situation arises when spatial compression is applied to a large volume of data gathered over a long time period. In the above cases, decreasing communication energy cost by compression is gained at the expense of computation cost for compression. Thus, maximum compression might not always lead to minimal energy

cost. Alternative methods for performing data compression need to be exploited such that the computation cost can be efficiently traded against the communication cost.

Towards the above purpose, we use the technique of tunable compression described in Section 2.2.3 to achieve balanced computation and communication costs for information routing. Specifically, we study the problem of constructing a data gathering tree spanning a set of source nodes and determining the flow from each source node to the sink with the goal of minimizing the sum of both computation and communication energy costs over all nodes in the tree. We refer to our problem as the Tunable Data Gathering (TDG) problem. Consider one specific case of our problem that has free computation and every node can always compress all of its incoming data to one unit data. Since such a case is exactly the Minimal Steiner Tree problem, the TDG problem is NP-Hard in general.

When lossy compression is considered, techniques such as the JPEG compression algorithm typically involve three stages: sampling, scalar quantization, and lossless binary encoding. By tuning parameters related to the effective sampling size and the quantization scaling, prior works have studied the tradeoffs between the quality of the compressed image and the computation and communication energy costs [114]. Hence, our work also complements these works by further exploring tradeoffs between computation and communication energy at the stage of lossless binary coding.

We consider the problem of constructing a data gathering tree spanning a set of source nodes and rooted at a sink node. For this problem, two important data compression schemes have been previously investigated in literature: distributed source coding [111] and compression with explicit communication [32]. When distributed source coding such as Slepian-Wolf coding is employed, the source data can be coded (compressed) without explicit communication among sources. Indeed, the study in [32] shows that Shortest Path Tree (SPT) is the optimal routing structure for such a coding scheme. However, such a theoretical limit has not been achieved by

any practical distributed source coding schemes yet. Hence, we focus on compression schemes with explicit communication — to perform joint data compression requires the availability of side information from other sources via explicit communication.

While most prior works on data gathering focus on minimizing the communication cost only, our distinguishing goal is to minimize the sum of both computation and communication costs by utilizing tunable compression. We refer to our problem as the Tunable Data Gathering (TDG) problem. To facilitate the tuning of compression over the data gathering tree, we propose a flow based model where data from each source is compressed and transmitted as a data flow over the corresponding path from the source to the sink. Hence, the TDG problem involves two related subproblems: to construct a data gathering tree and to determine the flow over all paths in the tree. Consider the special case of our problem that has free computation and every node can always compress all of its incoming data to one unit of data. Since such a case is exactly the Minimal Steiner Tree (MST) problem, the TDG problem is NP-Hard in general.

5.1.1 Our Contributions

We handle the TDG problem by decoupling tree construction and flow determination. We first show how the optimal flow can be determined for a given tree structure. By assuming a grid deployment of sensor nodes, we then model and analyze and performance of two existing tree construction methodologies, namely the Shortest Path Tree (SPT) and the Minimum Steiner Tree (MST). The results indicate that while SPT performs well when the relative computation cost compared with communication cost is high, MST is preferred when the relative computation cost is low. More importantly, MST provides a constant-factor approximation for the grid deployment throughout variations of the relative computation cost.

We also examine the performance of approximated MST (A-MST)[1] and SPT for general graphs through simulation. Our results further reveal the tradeoffs between A-MST and SPT with respect to several key system parameters, including relative computation cost, number of source nodes, and communication radius. Moreover, A-MST demonstrates acceptable average performance in the studied scenarios, which leads to the conclusion that A-MST is suitable as a practical solution due to its simplicity. For theoretical completeness, we also present a randomized tree construction methodology that achieves poly-logarithmic approximation for general graphs.

5.1.2 Chapter Organization

The related work is briefly discussed in Section 5.2. In Section 5.3, we give assumptions and models of our problem, which is formally defined in Section 5.4. The performance of SPT and MST on a grid deployment is analyzed in Section 5.5. The randomized approximation algorithm is then described in Section 5.6. Simulation results are presented in Section 5.7. Finally, concluding remarks are given in Section 5.8.

5.2 Related Work

The problem of constructing energy-efficient data gathering tree in wireless sensor networks while considering data compression is gaining increasing research attention. A nice description of several practical schemes for tree construction is presented in [87]. Using the terminologies from [87], these schemes include routing-driven compression (RDC), compression-driven routing (CDR), and cluster based routing. Essentially, RDC involves opportunistic data compression over an SPT; CDR performs maximum possible compression using a MST-like routing among source nodes before routing to the sink; and cluster based routing is a hybrid scheme of RDC

[1] This is because MST is NP-Hard for general graphs.

and CDR. The performance metric is the accumulated number of bits transmitted over each hop. Assuming that each source node generates one unit size of information, the study is motivated by two extreme cases. In the case of zero data correlation, by selecting a shortest path from each source node to route its data to the sink node, the optimal solution is the SPT. In the case of perfect data aggregation, exactly one unit size of data shall be routed on each edge of the tree, implying that the optimal solution is the MST. In between the above two extreme cases, the optimal tree structure shall resemble a hybrid scheme of SPT and MST. Since data correlation is usually high within a small area, a natural cluster-based scheme is to use a MST structure within each cluster and a SPT to route compressed data from each cluster to the sink. While the optimal cluster size depends on the correlation factor, a surprising result is that in a grid-based scenario, a near-optimal cluster size can be analytically determined purely based on the network topology and is insensitive to the correlation factor.

The idea of hybrid routing scheme in the previous section is conformed by several other results, under various assumptions to model the data correlation. Cristescu *et al.* assumes a simplified compression model [32], where the aggregation factor of a piece of information depends only on the availability of side information. Assume that each sensor node in the network generates one unit size of information. Whenever there is a side information transported to a source node, no matter the sources and size of this side information, the output of the source node after joint compression with the side information is a fixed value $\rho \in (0, 1]$. To minimize the cost of a routing tree, we need to minimize the cost of routing information from all internal nodes to the sink, which prefers a SPT structure. On the other hand, we also need to minimize the cost of routing side information from a leaf node to the set of internal nodes that utilize this side information and the sink, which favors a Minimal Spanning Tree. Hence, the optimal solution lies between a SPT and a Minimal Spanning Tree. In fact, the Shallow Light Tree (STL)

proposed by Bharat-Kumar *et al.* [19] can be used to provide a constant factor approximation of the considered problem.

While STL provides an approximation for both SPT and Minimal Spanning Tree, the situation becomes different if the source nodes are a subset of the sensor nodes instead of all sensor nodes. In this case, we need to construct a tree structure that simultaneously approximates both SPT and MST. Under a more general objective function that minimizes the sum of total cost of the tree and the accumulated length from each source node to the sink node where the cost and length can be defined based on two independent metrics, an approximation algorithms is proposed by Meyerson *et al.* [81] to achieve a $\log k$ performance bound, where k is the number of source nodes.

A simplified version of the algorithm in [81] is used to solve the problem of transporting information from a set of source nodes to the sink node when the joint entropy of a set of source nodes is assumed to be a concave, but unknown function of the number of source nodes [47]. The network topology is assumed to be a complete graph with shortest path distance being the edge cost, if necessary. The algorithm constructs a hierarchical matching tree using an iterative method that provides a $\log k$ approximation to the optimal solution, where k is the number of source nodes. It is noted that the assumption about concave data aggregation function essentially leads to an identical abstraction of the information routing problem with that of the single-source buy-at-bulk problem [9]. The key point is that the transmission cost spent on each edge is a concave function of the number of source nodes that use this edge to communicate to the sink. Another randomized algorithm for information routing on a grid of sensor nodes is proposed by Enachescu *et al.* [39], which is proved to have a constant approximation of the optimal performance.

While the problem studied in this chapter is motivated by the above work, it is distinguished in modeling computation energy as well as communication energy costs. To the best of the

authors' knowledge, this is the first piece of work that formally addresses such problems. This problem is important to study as more advanced and complex applications are being designed on sensor networks which would require increased computation complexity over large volume of data.

Although we still model joint entropy to be a concave function of number of source nodes, the results in [47] and [9] cannot be directly applied to our problem. This is because when the computation energy is considered, the overall cost on each edge may not be a concave function of the number of sources using this edge to communicate to the sink. Our work shows that by using the notion of probabilistic metric approximation [14], a randomized algorithm gives an expected $O(\log^2 v)$ approximation solution, where v is the number of sensor nodes in the network. It is worth noting that the approximation bound can be further improved to $\log v \log\log v$ [15] or $\log v$ [41]. However, our major purpose is to illustrate the tradeoffs between SPT and MST, hence the results in [14] suffices.

When lossy compression is considered, the so-called *adaptive fidelity processing* [40, 110, 114] can be used to trade energy consumption with quality of the compressed data. Lossless compression does not present this luxury, as the data compression must respect the entropy of the source information. However, our techniques is easily extendible to lossy compression, provided that certain pre-specified distortion constraints need to be satisfied.

5.3 Models and Assumptions

5.3.1 Table of Notations

A list of notations used in this chapter is given in Table 5.1.

Table 5.1: Table of notations

$NG =< V, L >$	the graph representing the underlying network
R	set of source nodes, $R \subseteq V$
w_i	the weight of edge $L_i \in L$
$sink$	the sink node in V
δ_i	number of source nodes in a subtree rooted at $V_i \in V$
p_i	the path from $V_i \in R$ to $sink$
z_i	the last edge on p_i, i.e., edge on p_i that connects to $sink$
$a \preceq b$	a is a predecessor of b on a path
γ	relative computation energy cost
f, f_e^u	flow, or a specific flow from node u on edge e
$g(s, f)$	the computation energy for compressing input data of size s to an output of size f
H_i	joint entropy of $i \geq 1$ unit data
ρ	date entropy rate, i.e., $\rho = H_1$
B_i	lower bound of compressing unit data with $i-1$ pieces of side information, defined as $B_i = \frac{H_i}{i}$
β_i	lower bound of flow at node v_i, i.e., $\beta_i = L_{\delta_i}$
W_i	the length of a path from v_i to $sink$
M, N	metric spaces on node set V
$d_M(u_1, u_2)$	distance between nodes u_1 and u_2 on metric space M
α	approximation factor
p	number of source nodes in our analytical grid deployment

5.3.2 Network Model

The underling network is modeled as an arbitrary network topology (Section 2.1.2): a graph representation $NG =< V, L >$ is used to abstract the underlying network with v sensor nodes and l communication links between nodes. We assume a simplified communication mechanism with a medium access control (MAC) protocol that ensures no packet collisions or interference in the network (which is commonly assumed by, e.g., [87, 32, 47]). As described in Section 2.1.2, the communication cost over each link is simply abstracted by a scalar valued weight associated with the link that indicates the energy cost of sending a data packet of unit size over the link. Let w_{L_i} or simply w_i denote the weight of link L_i. Let $sink \in V$ denote the sink node and $R \subseteq V$ denote the set of source nodes.

A data gathering tree is a subtree of NG rooted at $sink$ that contains R, denoted as $T =< V', L' >$, where $R \subseteq V' \subseteq V$ and $L' \subseteq L$. Let δ_i denote the number of source nodes in

the subtree rooted at V_i. Also, let p_i denote the path from V_i to $sink$, with $u \in p_i$ ($e \in p_i$) signifying that node u (edge e) is along the path. Recall our definition in Section 2.1.1, for two nodes $V_1, V_2 \in V'$, $V_1 \preceq V_2$ indicates that V_1 is a predecessor of V_2. Similarly, for two edges $L_1, L_2 \in L'$, $L_1 \preceq L_2$ indicates that L_1 is a predecessor of L_2.

5.3.3 Flow-Based Data Gathering

Consider a data gathering tree over a sensor network. To clearly model the energy cost of data compression at each node, we model data transmission over the tree as a composition of different data flows from each source node to $sink$ (i.e., each path from a source node to $sink$ over the tree corresponds to a data flow over the path). Hence, the number of incoming packets (flows) to a given node equals the number of source nodes in its subtree. We assume that the total computation energy spent by the node is the sum of energy for compressing each individual incoming packet. Also, the output size for compressing each incoming packet is lower bounded by the joint entropy of all incoming packets, which will be described later.

Now consider an arbitrary path $p(u)$ in the tree from a source node u to $sink$. Let f_e^u denote the flow over $e \in p(u)$ and $z(u)$ denote the last edge in $p(u)$, i.e, the edge incident to $sink$ in $p(u)$. We assume that the total energy spent on data compression over the path $p(u)$ is determined by the flow on $z(u)$, i.e., the total energy cost for data compression over $p(u)$ is calculated as $\frac{\gamma}{f_{z(u)}^u}$.

Following the entropy model in [47] (which also effectively abstracts the entropy models in [87, 32]), we assume that the joint entropy of any i source nodes, H_i is a non-decreasing and concave function of i with $H_1 = \rho$, where $\rho \in (0, 1]$ is the entropy of one unit of data. We assume that the compression of i incoming data flows at node v can be performed in such a way that the lower bound for compressing each data flow equals $B_i = \frac{H_i}{i}$, with $B_1 = H_1 = \rho$.

In other words, we assume that when maximal compression is performed on i pieces of source information, the fraction of compressible data of each piece is the same.

From the above flow-based data gathering and joint entropy model, for any $e = (a, b) \in p(u)$, we have $f_e^u \geq B_{\delta_a} = \frac{H_{\delta_a}}{\delta_a}$ (recall that δ_a is the number of source nodes in the subtree rooted at a). We also assume that H_i has the property such that $B_i \geq B_{i+1}$ for $i > 1$. Hence, when a data packet is compressed and transmitted along $p(u)$, the lower bound of flow decreases as the packet approaches $sink$.

5.3.4 Discussion

First, our analysis is not restricted to the specific $g(f)$ in (2.7). In fact, while the energy characteristics of various compression algorithms have been studied in [12], to develop accurate models for abstracting the energy cost of tunable compression is still an open problem. We note that the tradeoffs between computation and communication energy costs essentially depend on the convexity of the total energy cost function, for instance, shown in (2.8). Hence, the requirement that energy cost is inversely proportional to compression ratio is in fact one nice example that leads to such a convexity. We expect other models to be investigated in this context.

Second, the above flow model naturally models the data (information) streaming from sources to the sink and facilitates the computation of energy cost of compression. We consider only energy cost under this flow model. Nevertheless, performance metrics such as delivery latency can be defined by virtually combining different outgoing flows from a node as a whole and assessing the resulting time cost accordingly.

Also, our problem is based on the simplified assumption that the joint entropy of any set of i sources is H_i and the flow from any of the i sources is lower bounded by B_i after joint compression. To incorporate other more sophisticate joint entropy models is part of our future

work. We have assumed H_i as a convex function of i. In the special case of a stationary Gaussian random field with independent sources, H_i grows linearly with i. Since joint data compression does not help reduce the data volume in such a case, SPT is the optimal tree structure. The optimal flow on the SPT can be determined by the techniques presented in Section 5.5.1.

Third, the assumption of determining the compression energy over a path solely based on the flow on the last edge in the path ignores the costs of possible decompression or re-compression at different nodes along the path. This assumption is justified by two reasons. First, based on the study in [12], techniques such as *gzip* consumes very little time for decompression compared to the cost of compression. Second, since the flow on a path decreases as it approaches the sink, the total compression energy along the path can be approximated by calculating the energy cost based on the flow on the last edge. Nevertheless, to model the computation cost more accurately is part of our future work.

Fourth, Since our problem is to minimize the overall energy cost, the receiving energy of sensor nodes can be easily incorporated into the TDG problem by adjusting the weight on edges.

5.3.5 An Example

To illustrate the above flow model, consider the data gathering tree given in Figure 5.1, where nodes V_1, V_5, V_6, and V_7 are source nodes, nodes V_2 and V_3 are relaying nodes, and V_4 is $sink$. Consider the path from V_1, denoted as $\{V_1, V_2, V_3, V_4\}$. Based on the structure of the tree, there is 1 source node in the subtree rooted at V_1, which is V_1 itself, 2 source nodes (V_1 and V_5) in the subtree rooted at V_2 and 3 source nodes (V_1, V_5, and V_6) in the subtree rooted at V_3. Hence, the lower bound of flow on the path can be calculated as $B_{\delta_{V_1}} = B_1 = H_1$ on link (V_1, V_2), $B_{\delta_{V_2}} = B_2 = \frac{H_2}{2}$ on link (V_2, V_3), and $B_{\delta_{V_3}} = B_3 = \frac{H_3}{3}$ on link (V_3, V_4). The path together with

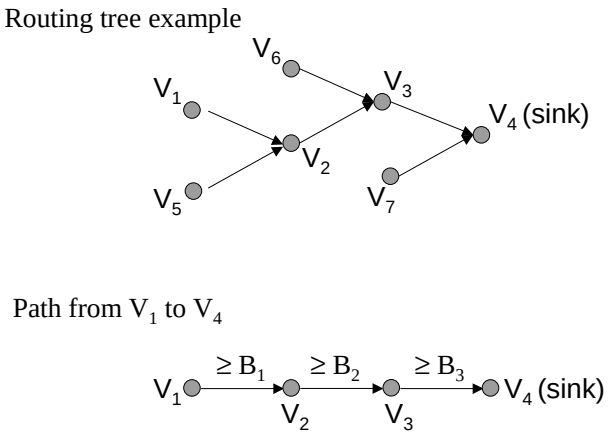


Figure 5.1: An example data gathering tree and a path within it

the lower bounds of flow on each link are also illustrated in Figure 5.1 (the superscription for f_i^V is omitted in the figure). Similarly, the flow on the path from V_5 to *sink* is lower bounded by B_1 on link (V_5, V_2), B_2 on link (V_2, V_3), and B_3 on link (V_3, V_4), respectively; the flow on the path from V_6 to *sink* is lower bounded by B_1 on link (V_6, V_3) and B_3 on link (V_3, V_4), respectively; the flow on the link from V_7 to *sink* is lower bounded by B_1.

5.4 Problem Definition

Based on the models and notations in Section 5.3, our *Tunable Data Gathering (TDG)* problem is formally defined as:

Given:

(i) a weighted graph $NG =< V, E >$ *with weight* w_i *for each* $E_i \in E$, $sink \in V$, $R \subseteq V$,

(ii) an energy function for data compression characterized by parameter γ, *and*

(iii) the joint entropy of i *sources,* H_i *and hence* $B_i = \frac{H_i}{i}$;

find *a subtree* $T =< V', E' >$ *that contains sink and all* $v \in R$*, and flow from all* $v \in R$ *to sink, so as to minimize*

$$\sum_{V_i \in R} (\frac{\gamma}{f^i_{z(i)}} + \sum_{e \in p_i} f^i_e w_e) \tag{5.1}$$

subject to

$$\forall V_i \in R, \forall e = (a, b) \in p_i \quad \Rightarrow \quad f^i_e \geq B_{\delta_a} \ , \tag{5.2}$$

where δ_a is the number of source nodes in the subtree rooted at sensor node a.

5.5 Analytical Study of SPT and MST

We consider two interesting special cases of the TDG problem. In the first case, we assume $\gamma = \infty$ or $B_i = 1$. Either condition leads to the obvious solution that no compression is performed in the data gathering tree. We construct the Shortest Path Tree (SPT) of NG by combining the shortest weighted path from every node in R to *sink*. Clearly, SPT is the optimal tree construction in this case. In the second case, we assume $\gamma = 0$ and $B_i = \frac{1}{i}$. In other words, computation is free and the joint entropy of any arbitrary i source nodes is always one. Hence, the desired flow on all edges is equal to one. Apparently, MST gives the optimal tree construction in this case.

Obviously, certain tradeoffs exist between SPT and MST, which is the main subject of this section. The rationale behind our study is to decouple between the two ingredients of selecting the tree construction and determining the flow from each source node to the sink. Therefore, we first show the optimal tunable compression strategy for a given path.

5.5.1 Optimal Flow on A Given Tree

Given a data gathering tree and an arbitrary source node $u \in R$, consider the path from the source node to the *sink*. Without loss of generality, let $p = \{V_1, V_2, \ldots, V_k\}$ denote the path, where V_1 is the source node, k is the number of links along the path, and V_k denotes the *sink*. Let $\vec{f}$ denote a vector of flow along the path, i.e., $\vec{f} = \{f^1_{e_1}, \ldots, f^1_{e_{k-1}}\}$. Since we are considering the specific path from V_1, we omit the superscription of elements in vector $\vec{f}$ as well as e in the subscription. Hence, we use $\vec{f} = \{f_1, \ldots, f_{k-1}\}$ to denote the flow vector.

To simplify the notation, let β_i to denote the lower bound of f_i, where $i \in [n-1]$. Since the path is extracted from a given tree, we can calculate β_i based on the structure of the tree (as shown by the example in Section 5.3.5). That is, $\beta_i = B_{\delta_{V_i}}$, where δ_{V_i} is the number of source nodes in the subtree rooted at V_i. Based on our model in Section 5.3.3, we have $\beta_i \leq \beta_{i-1}$. Also, let w_i denote the weight of $e_i = (V_i, V_{i+1})$, where $i \in [n-1]$. Let W_i denote the path length from e_i to e_{k-1}, i.e., $W_i = \sum_{j=i}^{k-1} w_j$. We slightly abuse the notation by letting $\beta_0 = 1$ and $W_k = 0$.

5.5.1.1 Example Revisited

We consider the flow on path V_1 to V_4 in Figure 5.1. Denote the flow as $\vec{f} = \{f_1, f_2, f_3\}$. Intuitively, when the relative computation cost increases, the optimal solution shall perform less amount of compression. In the trivial case when the computation cost is prohibitively high, no compression is performed and we have the optimal flow as $f_1 = f_2 = f_3 = 1$. Otherwise, the optimal flow can be obtained by examining the following three cases, depending on the relative cost of computation, which is reflected by γ.

1. The cost of compressing the input down to β_1 at node V_1 is more expensive than routing data of volume β_1 along the path. In this case, the optimal solution is to let V_1 compress the data to some $x \in [\beta_1, 1]$ and set $f_1 = f_2 = f_3 = x$.

2. Otherwise, another compression at node V_2 is necessary for reducing the total cost. If the cost of compressing the input at V_2 to β_2 is more expensive than the communication cost of routing β_2 over e_2 and e_3, the optimal solution is to set $f_1 = \beta_1$ and $f_2 = f_3 \in [\beta_2, \beta_1]$.

3. The compression is so cheap that it is also beneficial to perform one more compression at node V_3. In this case, the optimal flow is $f_1 = \beta_1$, $f_2 = \beta_2$, and $f_3 \in [\beta_3, \beta_2]$.

It can be understood that the optimal flow behaves as a piece-wise function of γ and w_i's, which abstracts the relative cost of computation. We assume that the weight of all edges The optimal flow for the above example is summarized in Table 5.2.

Table 5.2: Optimal flow for the example path

case	condition	optimal flow
1	$\gamma \geq W_1$	$f_1 = f_2 = f_3 = 1$
2	$\gamma < W_1$ and $\frac{\gamma}{\beta_1} \geq \beta_1 W_2$	$f_1 = f_2 = f_3 \in [\beta_1, 1]$
3	$\frac{\gamma}{\beta_1} < \beta_1 W_2$ and $\frac{\gamma}{\beta_2} \geq \beta_2 W_3$	$f_1 = \beta_1$, $f_2 = f_3 \in [\beta_2, \beta_1]$
4	$\frac{\gamma}{\beta_2} < \beta_2 W_3$	$f_1 = \beta_1$, $f_2 = \beta_2$, $f_3 \in [\beta_3, \beta_2]$

5.5.1.2 Determining the Optimal Flow

Based on the above intuition, we develop the following theorems for determining the optimal $\vec{f}$.

Lemma 2 *For any optimal flow $\vec{f}$ over the path p as previously described, if $f_{i+1} < f_i$, we have $f_i = \beta_i$.*

Proof: Otherwise, decreasing f_i to β_i does not change the cost for compression over p, since the compression energy is determined by the flow on the last link (V_{k-1}, V_k). However, this reduces the cost of communication over e_i since the flow over e_i is reduced to β_i, contradicting the optimality of the flow. ■

Theorem 2 *Given a path $p(v)$ as previously described, if $\gamma \geq W_1$, the optimal flow is of unit size on all links. Otherwise, suppose that $\gamma \in [W_{i+1}\beta_i^2, W_i\beta_{i-1}^2]$ for some $i \in [k-1]$, the optimal flow $\vec{f}$ is:*

$$\vec{f}(\gamma) = \{\beta_1, \beta_2, \ldots, \beta_{i-2}, \beta_{i-1}, \underbrace{f^*, \ldots, f^*}_{k-i}\}\ , \tag{5.3}$$

where $f^* = \max\{\beta_i, \sqrt{\frac{\gamma}{W_i}}\}$.

Proof: If $\gamma \geq W_1$, it means that any compression is more expensive than transmitting the original data along the path. Hence the optimal solution is to simply transmit the data packet without any compression. Otherwise, the proof is as follows.

First, since both W_i and β_i decreases with i, i.e., $W_{i+1} \leq W_i$ and $\beta_i \leq \beta_{i-1}$, the condition for γ is valid. Also, since $W_k\beta_{k-1}^2 = 0$ and $W_1\beta_0^2 = W_1$, the range of γ is within $[0, W_1]$.

Suppose that $\gamma \in [W_{i+1}\beta_i^2, W_i\beta_{i-1}^2]$ for some $i \in [k-1]$. Suppose that $\vec{x} = \{x_1, \ldots, x_{k-1}\}$ is the vector of the optimal flow with cost ϵ_x. Let f^* denote $\sqrt{\frac{\gamma}{W_i}}$. Let $\vec{f}$ denote the flow constructed by setting $f_j = x_j$ for $1 \leq j < i$ and $f_j = f^*$ for $i \leq j \leq k-1$. Let ϵ_f denote the cost of $\vec{f}$. We have

$$\begin{aligned}
\epsilon_x - \epsilon_f &= (\frac{\gamma}{x_{k-1}} + \sum_{j=1}^{k-1} x_j w_j) - (\frac{\gamma}{f^*} + \sum_{j=1}^{i-1} x_j w_j + f^* \sum_{j=i}^{k-1} w_j) \\
&= \frac{\gamma}{x_{k-1}} + \sum_{j=i}^{k-1} x_j w_j - (\frac{\gamma}{f^*} + f^* W_i) \\
&\leq (\frac{\gamma}{x_{k-1}} + x_{k-1} W_i) - (\frac{\gamma}{f^*} + f^* W_i)\ .
\end{aligned}$$

We define an optimization problem, $P(y)$, as to:

$$\begin{aligned}
\min \quad & \epsilon(y) = \frac{\gamma}{y} + yW_i \\
\text{subject to} \quad & y \geq \beta_i\ .
\end{aligned}$$

It is easy to verify that $P(y)$ is a convex function. We consider two cases for $\gamma \in [W_{i+1}\beta_i^2, W_i\beta_{i-1}^2]$.

Case (i): When $\gamma \in [W_i\beta_i^2, W_i\beta_{i-1}^2]$, we have $\sqrt{\frac{\gamma}{W_i}} \geq \beta_i$, hence implying $f^* = \sqrt{\frac{\gamma}{W_i}}$. Also, we have $\epsilon'(\beta_i) = -\frac{\gamma}{\beta_i^2} + W_i \leq 0$ and $\epsilon'(\beta_{i-1}) = -\frac{\gamma}{\beta_{i-1}^2} + W_i \geq 0$, where $\epsilon'(\cdot)$ is the first derivative of $\epsilon(\cdot)$. Therefore, the optimal y that leads to $\epsilon'(\cdot) = 0$ lies within $[\beta_i, \beta_{i-1}]$. By solving $\epsilon'(\cdot) = 0$, we know that the optimal value of y actually equals f^*. Thus we have $\epsilon_x - \epsilon_f \geq 0$, implying that $\vec{f}$ is actually optimal. From Lemma 2 and the fact $f^* \in [\beta_i, \beta_{i-1}]$, we have $f_j = \beta_j$ for $1 \leq j < i$.

Case (ii): When $\gamma \in [W_{i+1}\beta_i^2, W_i\beta_i^2]$, we have $\sqrt{\frac{\gamma}{W_i}} \leq \beta_i$, implying $f^* = \beta_i$. Also, we have $\epsilon'(\beta_i) = -\frac{\gamma}{\beta_i^2} + W_i \geq 0$. This means that $P(y)$ is an increasing function when $y \geq \beta_i$. Hence, the value of y that minimizes $P(y)$ is actually β_i. Again in this case, we have $\epsilon_x - \epsilon_f \geq 0$, implying that $\vec{f}$ is also optimal.

Finally, we can combine the above two cases using a max function for f^*. ■

From Theorem 2, the optimal flow is trivial when $\gamma \geq W_1$. Hence, we assume $\gamma < W_1$ in the following discussion. Theorem 2 reveals the fact that for a given value of γ that is within $[W_{i+1}\beta_i^2, W_i\beta_{i-1}^2]$ for some $i \in [k-1]$, the optimal flow on the last $k-i$ edges remains the same. In fact, we can eliminating the max function in the theorem by setting f^* to β_i, while the ratio of the energy cost of the resulting flow over the optimal cost is bounded by a constant, as shown by the following lemma.

Lemma 3 *Given a path p from V_1 to sink as previously described, suppose that where $\gamma \in [W_{i+1}\beta_i^2, W_i\beta_{i-1}^2]$ for some $i \leq k-1$. Define a flow $\vec{f}$ as $\vec{f} = \{\beta_1, \beta_2, \ldots, \beta_{i-1}, \underbrace{\beta_i, \ldots, \beta_i}_{k-i}\}$. The cost of $\vec{f_v}$ over the optimal is bounded by $\frac{\beta_{i-1}+\beta_i}{2\beta_i}$.*

Proof: We only need to consider the case when $\vec{f}$ differs from the optimal flow, which is Case (i) in the proof of Theorem 2. In this case, we have $W_i l_i^2 \leq \gamma \leq W_i l_{i-1}^2$. Let ϵ_{opt} denote the optimal cost and ϵ_f denote the cost of $\vec{f}$. Let $A = \sum_{j=1}^{i-1} \beta_j w_j$. We have

$$\begin{aligned} \epsilon_{opt} &= \frac{\gamma}{\sqrt{\frac{\gamma}{W_i}}} + A + W_i\sqrt{\frac{\gamma}{W_i}} \\ &= A + 2\sqrt{\gamma W_i} \\ \epsilon_f &= \frac{\gamma}{\beta_i} + A + \beta_i W_i \end{aligned}$$

Hence, the ratio of ϵ_f over ϵ_{opt} can be bounded as:

$$\begin{aligned} \frac{\epsilon_f}{\epsilon_{opt}} &= \frac{\frac{\gamma}{\beta_i} + A + \beta_i W_i}{A + 2\sqrt{\gamma W_i}} \leq \frac{\gamma + W_i\beta_i^2}{2\beta_i\sqrt{\gamma W_i}} \\ &= \frac{\sqrt{\gamma}}{2\beta_i\sqrt{W_i}} + \frac{\beta_i\sqrt{W_i}}{2\sqrt{\gamma}} \leq \frac{1}{2}\frac{\beta_{i-1}}{\beta_i} + \frac{1}{2} \\ &= \frac{\beta_{i-1} + \beta_i}{2\beta_i} \end{aligned}$$

■

Hence, given a data gathering tree T of NG, we can determine the optimal flow over T by applying Theorem 2 to each path in T, or get a constant factor approximation by applying Lemma 3.

Moreover, let $Diam(sink, R)$ denote the weighted diameter of NG with respect to R and $sink$, i.e., the maximum among the shortest weighted path from any node in R to $sink$. From Theorem 2, we have:

Corollary 1 *Given NG, if $\gamma \geq Diam(sink, R) \times L_1^2$, the Shortest Path Tree is the optimal tree for the TDG problem, with the flow specified by Theorem 2.*

Proof: From Theorem 2, the optimal flow from any $V_i \in R$ along its shortest path to $sink$ equals some fixed value within $[1, L_1]$ for all edges along the path. Thus, joint compression of

data from V_i with side information from other sources only decreases the lower bound of the feasible flow, but never reduces the optimal cost of the tree. ■

Let $\gamma^* = Diam(sink, R)L_1^2$. We refer to γ^* as the *critical point* of the system.

5.5.2 The Performance Bound in A Grid Deployment

For analytical tractability, we assume a grid deployment of size $r \times 2r$, where r source nodes at the leftmost column needs to send information to the *sink* located at the bottom right corner of the grid (similar network deployment has also been studied in [87, 97] for tractable analysis). Each sensor node can communicate to its one hop neighbors (i.e., 8 neighbors when ignoring boundary effects). We also assume $w_e = 1$ for all $e \in E$.

The routing constructed by SPT and MST are illustrated in Figure 5.2. Note that although to find an MST for a general graph is NP-Hard, the MST for the specific grid deployment in Figure 5.2 is trivial. From Corollary 1, the SPT is optimal when $\gamma \geq \gamma^* = (2r-1)L_1^2$. Hence, we are only interested in the performance of the SPT and MST for $\gamma \in [0, \gamma^*]$.

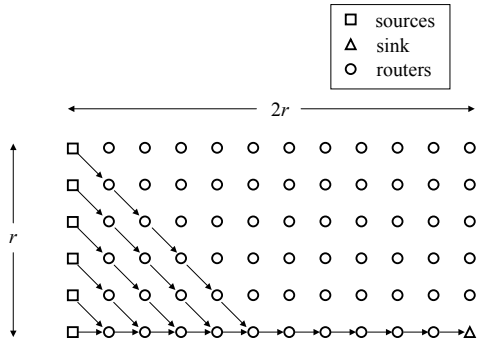
Let ϵ_{SPT} denote the energy cost for SPT and ϵ_{MST} for MST. Using Lemma 3, ϵ_{SPT} can be approximated as

$$\epsilon_{SPT} = \begin{cases} C + i(\frac{\gamma}{L_i} + L_i(2r-(i+1))) + \sum_{j=1}^{i} jL_j + \sum_{j=r}^{2r-i-1} (\frac{\gamma}{L_1} + L_1 j), \\ \qquad \text{for } (2r-i-1)L_{i+1}^2 \leq \gamma \leq (2r-i)L_i^2, \ \ i = 1, \ldots, r-1 \qquad (5.4a) \\ C + \frac{r\gamma}{L_r} + r^2 L_r + \sum_{j=1}^{r-1} jL_j, \qquad \text{for } 0 \leq \gamma \leq rL_r^2 \qquad (5.4b) \end{cases}$$

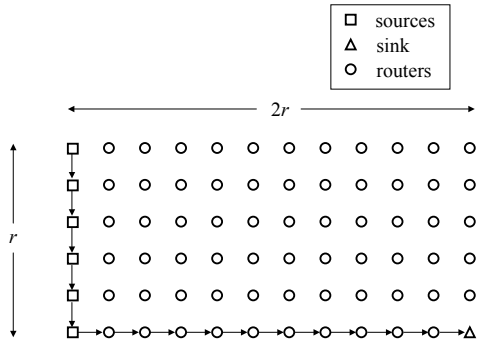
where $C = \frac{r(r-1)L_1}{2}$.

Let $q = 3r - 1$. Let i^* be the smallest integer such that $(q - i^*)L_{i^*}^2 \geq \gamma^*$. We can also approximate ϵ_{MST} as follows:

$$\epsilon_{MST} = \begin{cases} 2\sqrt{\gamma}\sum_{j=2r-1}^{q-i-1}\sqrt{j} + \sum_{j=1}^{i} jL_j + i(\frac{\gamma}{L_i} + L_i(q-i-1)), \\ \qquad\qquad \text{for } (q-i-1)L_{i+1}^2 \leq \gamma \leq (q-i)L_i^2, \;\; i = 1, \ldots, i^* \quad (5.5a) \\ \frac{r\gamma}{L_r} + \sum_{j=1}^{r-1} jL_j + r(2r-1)L_r, \qquad \text{for } 0 \leq \gamma \leq (2r-1)L_r^2 \quad (5.5b) \end{cases}$$



(a) SPT



(b) MST

Figure 5.2: SPT and MST routing schemes

Moreover, the minimal cost of the TDG problem is lower bounded by replacing constraint (5.2) with $\forall V_i \in R, \forall e \in p_i, f_e^i \geq B_{|R|}$, where $|R| = r$ in this particular case. In

other words, we assume that distributed source coding among source nodes is available at no extra cost. It can be verified that the optimal routing for such an lower bound case forms a SPT. Hence, the energy costs for the lower bound, ϵ_{LB} can be modeled as

$$\epsilon_{LB} = \begin{cases} 2r\sqrt{\gamma(2r-1)}, & \text{for } (2r-1)L_r^2 \le \gamma \le \gamma^* \quad (5.6a) \\ \dfrac{r\gamma}{L_r} + r(2r-1)L_r, & \text{for } 0 \le \gamma \le (2r-1)L_r^2 \quad (5.6b) \end{cases}$$

Based on the above results, we make the following observation about the performance bound of SPT and MST with respect to the above lower bound on the specific grid deployment.

Observation 1 *For the grid deployment in Figure 5.2, we have the following performance bound regarding SPT and MST:*

$$\lim_{\gamma \to \gamma^*} \frac{\epsilon_{SPT}}{\epsilon_{LB}} = O(1) \quad (5.7)$$

$$\lim_{\gamma \to 0} \frac{\epsilon_{SPT}}{\epsilon_{LB}} = O(\frac{r}{H_r}) \quad (5.8)$$

$$\lim_{\gamma \to \gamma^*} \frac{\epsilon_{MST}}{\epsilon_{LB}} = O(1) \quad (5.9)$$

$$\lim_{\gamma \to 0} \frac{\epsilon_{MST}}{\epsilon_{LB}} = O(1) \quad (5.10)$$

where $\gamma^* = (2r-1)L_1^2$ *is the critical point of the system.*

Proof: From (5.6a) and the fact $\gamma^* = (2r-1)L_1^2 = \Omega(r)$, we have

$$\lim_{\gamma \to \gamma^*} \epsilon_{LB} = \Omega(r\sqrt{\gamma r}) = \Omega(r^2) \text{ and}$$

$$\lim_{\gamma \to 0} \epsilon_{LB} = \Omega(r^2 L_r) = \Omega(rH_r) .$$

Equation (5.7) directly follows from Corollary 1. To prove (5.8), we bound the cost of SPT based on (5.4b). From the condition in (5.4b), we have $\frac{\gamma}{L_r} \leq rL_r = H_r$ and thus

$$\begin{aligned} \lim_{\gamma \to 0} \epsilon_{SPT} &\leq C + 2rH_r + \sum_{j=1}^{r-1} H_j \\ &= C + O(rH_r) \ , \end{aligned}$$

where $C = \frac{r(r-1)L_1}{2} = O(r^2)$. Hence, (5.8) follows.

From the condition in (5.5a), we have $\frac{\gamma}{L_i} \leq (q-i)L_i$. Since $\gamma^* = (2r-1)L_1^2$, we have

$$\begin{aligned} \lim_{\gamma \to \gamma^*} \epsilon_{MST} &= O(\sqrt{r} \sum_{j=2r-1}^{q-i-1} \sqrt{j}) + \sum_{j=1}^{i} H_j + O(i(q-i)L_i) \\ &= O(r^2) + \sum_{j=1}^{i} H_j + O((q-i)H_i) \\ &= O(r^2) + O(rH_r) \ . \end{aligned}$$

Thus,

$$\begin{aligned} \lim_{\gamma \to \gamma^*} \frac{\epsilon_{MST}}{\epsilon_{LB}} &= \frac{O(r^2) + O(rH_r)}{\Omega(r^2)} \\ &= O(1) + O(\frac{H_r}{r}) = O(1) \end{aligned}$$

Lastly, we bound the cost of $\lim_{\gamma \to 0} \epsilon_{MST}$. From the condition in (5.5b), we have $\frac{\gamma}{L_r} \leq (2r-1)L_r$. Hence,

$$\begin{aligned} \lim_{\gamma \to 0} \epsilon_{MST} &\leq \sum_{j=1}^{r-1} H_j + (2r-1)rL_r \\ &= O(rH_r) \end{aligned}$$

and (5.10) follows. ■

Though the results in Observation 1 is derived for the grid deployment in Figure 5.2, it gives an insight into why MST may perform well even in general graph regardless of the form of H_i and the relative energy cost γ.[2] Although MST is NP-hard on a general graph, our simulation results suggest that even an approximated MST can be used as a practical routing scheme for solving the TDG problem.

5.5.3 Tradeoffs Between SPT and MST

For the purpose of demonstration, we instantiate H_i based on the joint entropy model from [78]. Specifically, for a stationary Gaussian random process with correlation coefficient as e^{-d^2} and a scalar quantizer with uniform step size and infinite number of levels, H_i scales as $O(\log i)$ as $i \to \infty$. Hence, we set $H_i = \rho \log i$, where ρ is the data entropy rate. According to the lossless compression ratio for CCITT test images [27], we set $\rho = 0.1$. In Figure 5.3, we plot ϵ_{SPT}, ϵ_{MST}, and ϵ_{LB} for $r = 20$, $\rho = 0.1$ and varying γ between 0 and $\gamma^* = (2r-1)\rho^2 = 0.39$.

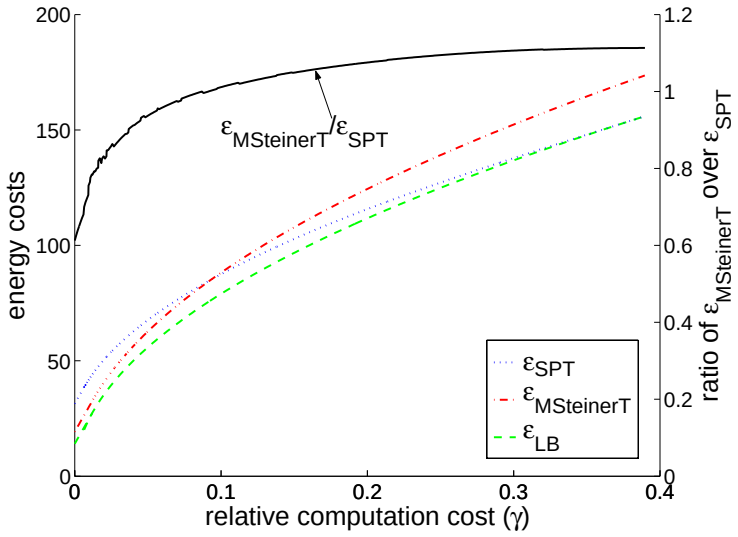


Figure 5.3: Performance of SPT and MST for grid deployment

[2] Theoretically, the performance of MST for the TDG problem in general graphs is unbounded in the worst case.

From Figure 5.3, we can clearly observe the tradeoffs of the performance of SPT and MST with respect to variations in γ. When γ is large, SPT outperforms MST with ϵ_{SPT} approaching to ϵ_{LB}. This is because large computation cost discourages data compression, hence shortest paths from source nodes to sink are preferred for saving communication costs. However, the performance of MST is also quite satisfactory when $\gamma = 0.4$, with no more than 15% increase over SPT.

On the other hand, when γ approaches to zero, MST provides a nearly 40% reduction with respect to SPT. Essentially, when the computation costs is low, compressing data from multiple sources before routing to the sink gains great advantage in reducing the flow on the tree and hence the communication costs. In the special case of $\gamma = 0$, our problem becomes similar to the scenario studied in [87], where tradeoffs between MST and SPT exist due to variations in spatial correlation (essentially captured by H_i). Briefly, results in [87] state that MST outperforms SPT when the spatial correlation among source nodes is high and SPT outperforms MST when the spatial correlation is low. In our study, the specific form of H_i determines that MST performs better than SPT, which is in keeping with the results presented in [87].

Though the above analysis is based on a specific grid topology, it clearly shows that MST provides reasonably good performance with respect to variations in γ. Another important insight is that the optimal solution to TDG shall explore the tradeoffs between SPT and MST when γ varies. This is not surprising since γ essentially abstracts the cost of compression. Hence, the extent to which routing shall be driven by compression depends on γ. The above analysis suggests a tree structure that resembles both SPT and MST for solving the TDG problem. In the next section, we show the use of a hierarchically clustered tree for this purpose.

5.6 A Randomized $O(\log^2 v)$ Approximation

In this section, we show a randomized algorithm with an expected $O(\log^2 v)$ approximation performance based on the k-hierarchically well-separated tree (k-HST) proposed in [14] (recall that v is the number of vertices in NG). The key idea is to approximate the graph NG with a set of k-HST's such that the routing selected according to a randomly chosen k-HST is expected to have a cost at most $O(\log^2 v)$ times the optimal.

We first give the notion of probabilistic metric approximations from [14]. Let V denote a set of n points and M a metric space over V where the distance between $u_1 \in V$ and $u_2 \in V$ is denoted by $d_M(u_1, u_2)$.

Definition 10 *A metric space N over V, dominates a metric space M over V, if for every $u_1, u_2 \in V$, $d_N(u_1, u_2) \geq d_M(u_1, u_2)$.*

Definition 11 *A metric space N over V, α-approximates a metric space M over V, if it dominates M and for every $u_1, u_2 \in V$, $d_N(u_1, u_2) \leq \alpha \cdot d_M(u_1, u_2)$.*

Definition 12 *A set of metric spaces $\mathcal{S}$ over V, α-probabilistically-approximates a metric space M over V, if every metric space in $\mathcal{S}$ dominates M and there exists a probability distribution over metric spaces $N \in \mathcal{S}$ such that for every $u_1, u_2 \in V$, $E(d_N(u_1, u_2)) \leq \alpha \cdot d_M(u_1, u_2)$.*

Definition 13 *A k-hierarchically well-separated tree (k-HST) is defined as a rooted weighted tree with the following properties:*

- *The edge weight from any node to each of its children is the same.*
- *The edge weights along any path from the root to a leaf are decreasing by a factor of at least k.*

In the above definition, $k > 1$ is a pre-specified constant. The main results in [14] can be simply stated as follows:

Theorem 3 *Every weighted connected graph NG can be α-probabilistically-approximated by a set of k-HST's constructed from NG, denoted as $\mathcal{S}$, where $\alpha = O(\log^2 v)$. Moreover, the probability distribution over $\mathcal{S}$ is computable in polynomial time.*

The construction of the k-HST's is based on a randomized recursive partitioning algorithm. Regarding NG as a cluster of nodes, the algorithm starts by randomly partitioning NG into a set of sub-clusters, with each sub-cluster having a diameter at most $1/k$ of the diameter of NG. A set of nodes are then created for NG and each of the sub-clusters. These nodes form a tree rooted at the node created for NG with all the edge weights set to $1/k$ of the diameter of NG. The above procedure is recursively performed for each sub-cluster till each sub-cluster contains only one node from NG. Details of the construction of the set of k-HST's can be found in [14].

Using the above metric approximations, we can show:

Theorem 4 *Given a TDG problem on graph NG with optimal cost equal to C, there is a feasible solution on the set of k-HST's that α-probabilistically-approximate NG with the expected cost (over the distribution on the k-HST's) to be at most αC.*

Proof: Consider an arbitrary path from the optimal tree to the TDG problem, denoted as T^*. Without loss of generality, let let $p = \{V_1, V_2, \ldots, V_k\}$ denote the path from V_1 to $sink$, where V_1 is the source node, k is the number of links on the path, and V_k is the $sink$. We still use the notations in Theorem 2. Let f_j denote the flow on link $e_j = (V_j, V_{j+1})$, where $j \in [k-1]$. From Theorem 2, the cost of p, $C(p)$ can be calculated as:

$$C(p) = \sum_{j=1}^{i-1} w_j \beta_j + \frac{\gamma}{f^*} + f^* \sum_{j=i}^{k-1} w_j \ ,$$

where $\gamma \in [W_{i+1}\beta_i^2, W_i \beta_{i-1}^2]$ for some i, and $f^* \in [\beta_i, \beta_{i-1}]$.

Consider any edge $e_j = (V_j, V_{j+1})$ on p. We associate with the edge, in an arbitrary tree $M \in S$, a path M_{e_j} between V_j and V_{j+1} of length $d_M(e_j) = w_i \alpha_M$, where $E(\alpha_M) = O(\alpha)$.

Thus, the path p is associated with a path in M, denoted as $p_M = \{M_{e_1}, M_{e_2}, \ldots, M_{e_{k-1}}\}$. In the following, we construct a feasible flow on p_M and bound the cost of this flow by $O(\alpha)$ times the cost of p. Note that the path p_M may not be simple, but this does not affect the construction of a flow on the path and the calculation of the cost of the flow.

For each path in $M_{e_j} \in M$ that corresponds to an edge in $e_j \in T^*$, let β'_k denote the lower bound of the flow over each link e'_k along M_{e_j}. Now consider the optimal tree T^* and the tree composed by all M_{e_j}'s, denoted as T_M. Since the number of source nodes that have their flow going through e'_k in T_M cannot be less than the number of flows going through e_j in T^*, we have $\beta'_k \leq \beta_j$. Recall that f_j is the flow on edge e_j over p in the optimal solution. Thus, setting the flow on each edge of M_{e_j} to be f_j gives a feasible flow on p_M. Moreover, the expected cost of this flow can be calculated as:

$$\begin{aligned} C(p_M) &= \sum_{j=1}^{i-1} d_M(e_j)\beta_j + \frac{\gamma}{f^*} + f^* \sum_{j=i}^{k-1} d_M(e_j) \\ &\leq \alpha_M \sum_{j=1}^{i-1} w_j\beta_j + \frac{\gamma}{f^*} + \alpha_M f^* \sum_{j=i}^{k-1} w_j \\ &\leq \alpha_M C(p) \end{aligned}$$

where $\gamma \in [W_{i+1}\beta_i^2, W_i\beta_{i-1}^2]$ for some i, and $f^* \in [\beta_i, \beta_{i-1}]$.

The above result indicates that for each path in T^*, we can construct a feasible path in an arbitrary $M \in S$, with the cost of the path to be bounded by α_M. Since the cost of T^* is the sum of $C(p)$ over all nodes in R and $E(\alpha_M) = O(\alpha)$, the theorem is proved. ■

It is easy to show that the optimal solution to a TDG problem over a tree T is simply the composition of routes from each node in R to *sink* on T. Thus, by mapping each route in T to a path in NG, we can get the data gathering tree in NG. The problem with the mapping is that from the construction of T, every node in NG actually corresponds to a leaf in T, which means all internal nodes in T do not correspond to the nodes in NG. To handle this issue, we

simply map each internal node in T to an arbitrary node in the cluster corresponding to the internal node. From the fact that the weight of any edge incident from the internal node is $1/k$ of the diameter of the corresponding cluster, the possible increase of path length resulting from the above mapping is bounded by a factor of constant k.

Thus, we have the following algorithm:

1. Choose at random a tree $T \in \mathcal{S}$.
2. Map the route from each $v \in R$ to $sink$ on T to a path on NG from v to $sink$ as previously described.
3. Determine the optimal flow on each path based on Theorem 2.

Theorem 5 *Given a TDG problem on a graph G, the above randomized algorithm gives a $O(\log^2 v)$ approximation.*

5.7 Simulation Results

5.7.1 Simulation Setup

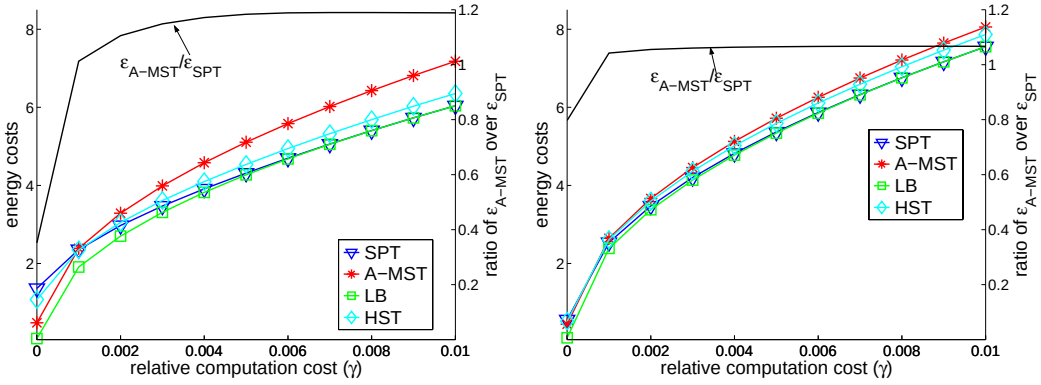
A sensor network was generated by randomly scattering v sensors in a unit square with a uniform distribution. The largest communication range of the radio devices was set to r, which in turn determines the number of neighbors for each sensor node to be around $v\pi r^2$ (ignoring boundary effect). The weight on each edge can be modeled by using various path-loss models in wireless communication. For our purpose, we set the weight on each edge to be the geometry distance between the two nodes incident to the edge. The sink node was always fixed at the left-bottom corner of the square, while the source nodes were randomly selected from all the sensor nodes in the square. We used the same joint entropy model as the one given in Section 5.5.3.

The performance of three tree construction methods, namely the SPT, MST, and the k-HST (or simply HST), is studied by simulation. While SPT and k-HST can be constructed based on

polynomial time algorithms, the construction of MST is NP-Hard for general graphs. We used the Greedy Incremental Tree (GIT) algorithm [68] that gives a 2-approximation MST, with A-MST denoting the resulting approximated MST. Moreover, the lower bound of the TDG problem were obtained using the relaxation method described in Section 5.5.2 with ρ fixed to 0.1.

All the data shown in this section is averaged over 300 instances such that they have a 95% confidence interval with a 5% (or better) precision. For each instance, the sensor field was randomly generated based on the above description.

5.7.2 Results



(a) Number of sensor nodes $n = 400$

(b) Number of sensor nodes $n = 50$

Figure 5.4: Impact of the number of sensor nodes n and the relative computation cost γ ($|R| = 30$, $r = 0.2$)

5.7.2.1 Impact of the number of sensor nodes n and the relative computation cost γ

For the results shown in Figure 5.4(a), we fixed $v = 400$, $|R| = 30$, $r = 0.2$, while varying γ within $[0, 0.01]$ (we have $\gamma^* = Diam(sink, R)L_1^2 \approx 0.014$). The first thing to notice is that

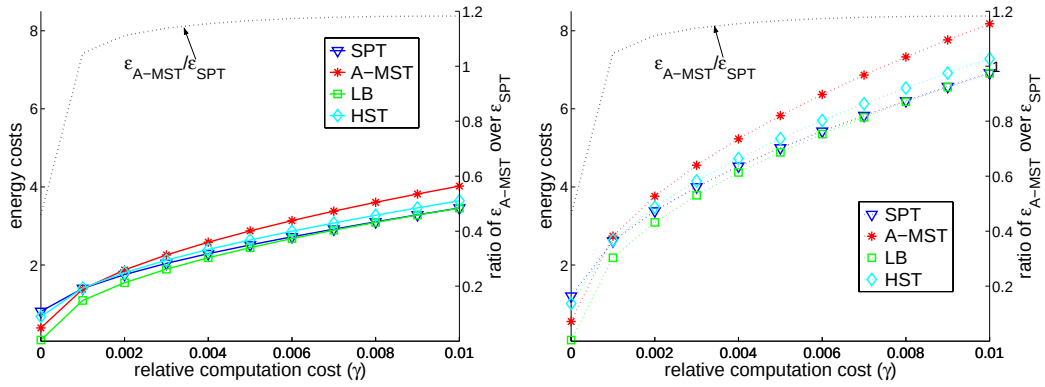
the simulation results in general graphs conform the analytical tradeoffs between SPT and A-MST described in Section 5.5.2. The performance of SPT approaches to the lower bound when γ is close to 0.1, while A-MST outperforms SPT when γ tends to zero. As expected, HST performs in between SPT and A-MST throughout the variations in γ. More importantly, A-MST demonstrates quite acceptable performance throughout the variations of γ. The curves of $\epsilon_{MST}/\epsilon_{SPT}$ clearly show that A-MST offers 60% energy savings over SPT for low γ, and less than 20% increase over SPT at high γ.

We also illustrate the results for $n = 50$ in Figure 5.4(b). We observe that the tradeoffs between SPT, A-MST, and HST still hold. Also, that the energy cost for each tree construction decreases with n when γ is large, since better routing can be discovered with more sensor nodes available in the field. However, because SPT always routes through the shortest path which may hinder data aggregation, the energy cost of SPT increases with n when $\gamma = 0$.

The lesson we learn from the results is that when the relative computation cost γ is known, either SPT or A-MST can be selected accordingly as a practical routing scheme. A-MST performs well on the average, with only slight degradation compared to SPT when γ is large. Although HST can be used to provide an approximation with guaranteed performance bound, the complexity involved in the implementation of HST does not gain much performance advantage over A-MST in the studied scenarios (although theoretically A-MST does not have a performance bound).

5.7.2.2 Impact of the number of source nodes $|R|$ and γ

For the results shown in Figure 5.5, we fixed $v = 200$, $r = 0.2$, while setting $|R|$ to 20 or 40 and varying γ within $[0, 0.01]$. It is quite understandable that the energy costs of all tree structures increase with $|R|$. Nevertheless, the tradeoffs between SPT, A-MST, and HST still hold for different $|R|$.



(a) Number of source nodes $|R| = 20$

(b) Number of source nodes $|R| = 40$

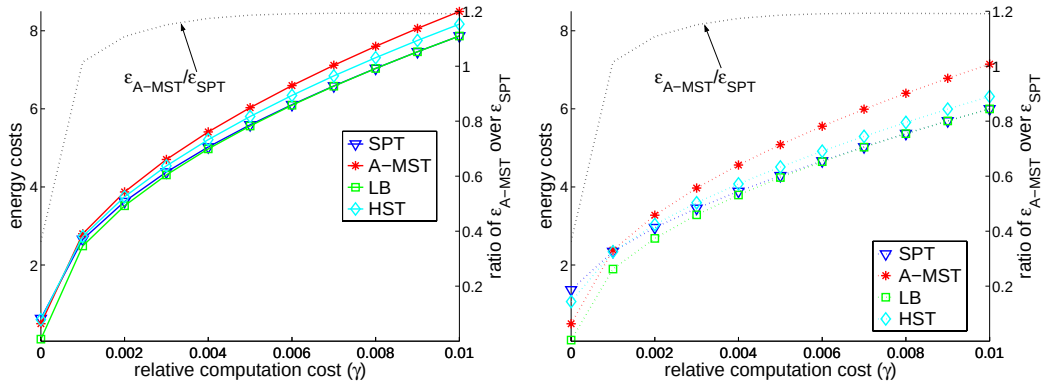
Figure 5.5: Impact of the number of source nodes $|R|$ and the relative computation cost γ ($n = 200$, $r = 0.2$)

5.7.2.3 Impact of the communication range r and γ

For the results shown in Figure 5.5, we fixed $v = 200$, $|R| = 30$, while setting r to 0.1 or 0.3 and varying γ within $[0, 0.01]$. The tradeoffs between SPT, A-MST, and HST are still apparent for different r. It can also be observed that increasing either r or n leads to very similar impact on the performance of different tree structures. This is because both result in increased number of neighbors, which in turn offers opportunities for better tree construction.

5.8 Concluding Remarks

In this chapter, we have studied the Tunable Data Gathering (TDG) problem of constructing a data gathering tree in wireless sensor networks while taking both computation and communication energy into consideration. Such problems are important for the development of advanced and complex applications for sensor networks that involve increasingly high computation energy cost. To facilitate the tradeoffs between computation and communication energy, we have applied the notion of tunable compression in the TDG problem.



(a) Communication range $r = 0.1$

(b) Communication range $r = 0.3$

Figure 5.6: Impact of the communication range r and the relative computation cost γ ($n = 200$, $|R| = 30$)

We have developed suitable energy model for tunable compression and a flow based model for gathering information from correlated sources. We have derived the optimal solution for scheduling tunable compression at different nodes on a given data gathering tree. Further, we have both analytically and empirically illustrated the tradeoffs between two data gathering trees, namely the Shortest Path Tree (SPT) and the Minimal Steiner Tree (MST). We showed that SPT performs close to optimal when the relative computation cost γ is high, while MST performs better when γ is low. Thus, the availability of such information would help the network designers to select the appropriate routing scheme. When information about γ is unknown or if it shows large application-specific spatio-temporal variations, a randomized algorithm can be used to provide a guaranteed poly-logarithmic approximation. However, we have shown that the simple (greedy) approximation of MST (A-MST) provides a constant-factor approximation for the grid deployment and very acceptable performance on the average for general graphs. Due to its simple implementation, the A-MST is attractive for practical applications.

Chapter 6

Concluding Remarks and Future Directions

In this chapter, we present concluding remarks of the thesis and a list of future research directions.

6.1 Concluding Remarks

We have presented techniques for three specific topics centered around cross-layer optimization for energy-efficient information processing and routing in wireless sensor networks. We summarize the three topics along four dimensions, including their functionalities, objectives, performance constraints, and the tradeoffs enabled by the investigated system knobs.

From the perspective of application level functionality, these three topics cover various stages of information processing and routing, including in-cluster information processing, information transportation over a given tree substrate, and the construction of a routing tree. As pointed out in Chapter 1, these three topics are particularly suitable for certain cluster-based network infrastructure [30, 52, 106, 130, 129]. In such an infrastructure, information sensing and proper information processing is performed in a localized fashion in each cluster while the output from all clusters are then routed to the base station. By addressing these various stages of

information processing and routing, we believe that our work provides a framework over which various extensions can be further overlaid. We are also aware of other schemes for information processing and routing, for instance, the network flow-based scheme [56, 62, 58]. To apply the presented optimization techniques in the context of such schemes are very interesting topics.

From the perspective of optimization objectives, we aim to balance the energy cost of sensor nodes for in-cluster processing or to minimize the overall energy costs for the other two topics on information transportation and routing. While both objectives can be used to improve the energy-efficiency of the system, their pros and cons still have not been well quantified by any research efforts at this time. In general, to balance the energy cost can be used to provide fairness in the lifetime of sensor nodes. By avoiding the so-called "hot spots" in the network, which consist of a set of heavily loaded sensor nodes that die earlier than other sensor nodes due to depleted energy, we gain advantages in terms of network connectivity and coverage. However, to balance the energy cost may not lead to a minimal overall energy cost, hence potentially reducing the lifetime of the system. Also, people can argue that in a highly dense network, energy-balance can be realized by other system knobs such as sleep scheduling. Therefore, it may be easier for just minimizing the total energy cost of awaking sensor nodes while still achieving certain extent of energy-balance across the system. Future works are needed to further justify the tradeoffs between the two objectives.

From the perspective of performance constraints, we have considered real-time latency constraint for both in-cluster processing and information transportation over a given tree substrate. Optimization under such a constraint is important for many mission-critical applications that are envisioned in the near future [77, 18, 21]. One of the unsolved issues is that the latency constraint for the above two operations is normally given as a whole from the user. It is not clear how to break the constraint into two parts that can be applied to the two operations respectively. In the topic of information transportation over a given tree, the impact of joint

information entropy is addressed by assuming a given aggregation function. Also, the constraint of joint information entropy is explicitly considered in the topic of constructing a routing tree. However, the latency constraint is not considered in this topic due to its high complexity. We also note that there are other performance metrics, including throughput and information fidelity that can be considered in future.

From the perspective of involved system knobs, we have presented an unified scheme for voltage scaling and rate adaptation of in-cluster processing to explore the energy-latency tradeoffs. A similar tradeoff is explored for information transportation over a given tree. Although not explicitly emphasized, the work on in-cluster processing also achieves certain balance between the computation and communication energy costs. Such a balance is explicitly addressed by using tunable compression in the problem of information routing. When other performance metrics are considered, we can imagine a larger tradeoff space by employing different system knobs accordingly, including those to be discussed in the next section.

6.2 Future Directions

There are several possible directions to extend the works presented in this thesis. We first discuss in detail the notion of adaptive fidelity algorithms, an interesting system knob that is closely related to the presented works. Then we will identify a set of future directions in a broad context of information processing and routing for sensor networks. These directions are chosen to address issues including node mobility, wireless communication reliability, and integration with existing technologies, respectively.

6.2.1 Adaptive Fidelity Algorithms

While adaptive fidelity algorithms were briefly mentioned in Chapter 5, we present a more detailed discussion here. According to the definition of Estrin *et al.* [40], "adaptive fidelity algorithm is one where the quality (fidelity) of the answer can be traded against battery lifetime, network bandwidth, or number of active sensors". An understanding of this definition at a higher level of abstraction is that the fidelity of the information delivered to end users is one of the application level performance metrics that can be traded against each other, including computation and communication capabilities and energy costs. Hence, by considering adaptive fidelity algorithm, we enlarge the design space of cross-layer optimization with one more dimension.

One example given by Estrin *et al.* [40] is that we can selectively switch off certain portion of sensor nodes in object localization. While the precision of the object location may be compromised because less sensor nodes are involved in the triangulation process, the energy costs of the localization is also reduced. Consider another example of JPEG in image compression. Studies have shown that by tuning two parameters — quantization level and Virtual Block Size (VBS) — it is possible to trade the image quality with both processing delay and compression output size [114]. While processing delay can be linearly translated to computation energy, the output size can also be translated to either bandwidth requirement or communication energy.

To apply adaptive fidelity algorithms into the context of information processing and routing in sensor networks involves several challenges. The major challenge is to identify the suitable system knobs to realize graceful tradeoffs between the information fidelity and the energy costs, which seems to be very application-specific.

One example is the so-called energy scalable algorithm proposed by Sinha *et al.* [110] that performs algorithmic transform such that a nice energy *vs* quality scaling can be achieved for a specific set of computation. Consider the simple example of calculating the sum of a list of

numbers. The key idea of the transform is to sort the vectors so that the numbers that might dominate the final sum are accumulated first. In case that the energy budget is not sufficient for performing all the required adding operations, the last several numbers that are least significant are ignored. By doing so, the accuracy of the final sum can be gracefully traded against the energy cost. Such a transform may be more complicated for other applications (please refer to [110] for several examples pertinent to real applications). Also, the overhead of the transform should be relatively small.

When image compression is concerned in video surveillance applications, the aforementioned two system parameters — quantization level and VBS — can be used to adjust the quality of the delivered image and both computation and communication energy costs. Intuitively, quantization level affects the precision of the sampling while VBS tunes the fraction of pixels that are sampled by the compression algorithm. Hence, we can either increase quantization level or decrease VBS to reduce the computation intensity and the output size of the compression, but with degraded image quality.

Another challenge lies in the inter-relationship between adaptive fidelity algorithms and other system knobs including the three that have been investigated in this thesis. For instance, it is not clear to us how the above algorithmic transform can be integrated with voltage scaling to achieve further reduced energy cost for performing a computation task with latency constraint and certain tolerance in the accuracy of the results. Also, it is not clear how the above coding techniques for image compression can be integrated into a system with channel coding techniques at wireless communication level (including rate adaptation) and application level tunable compression.

Moreover, it is understood that the integration of adaptive fidelity algorithms with other system knobs should also be quite application-specific. Hence, identify the set of applicable

system knobs for a specific application scenario is a crucial step (which in fact already holds in the broad context of cross-layer optimization).

6.2.2 Directions from a Broad View

6.2.2.1 Consideration for Mobile Sensor nodes

The works in the thesis are based on static networks where mobility of sensor nodes are not considered. Another trend in WSNs is the integration of mobile nodes into the traditionally static network [59]. The presence of mobile nodes will change the basic method for routing information across the network, implying a possible integration of existing results in mobile ad hoc networks together with coding techniques for correlated information source.

An interesting approach for information processing and routing in mobile WSNs is to use mobile nodes to transport and gather information from stationary nodes, which in most cases are sparsely dispersed. In such an approach, mobile nodes will affect the de-composition of system energy cost — a large portion of the system energy will be spent on moving the mobile nodes. Hence, a carefully scheduled mobility of nodes is crucial to maximize the system lifetime. Knowledge on information availability and correlation among sources can be used to assist the mobility scheduling so that energy is saved by avoiding visiting nodes with no new information.

Intuitively, various tradeoffs between the energy cost and the quality of the gathered information can be explored. In case of unlimited storage at all stationary nodes, the quality of the information can be captured by the timeliness of the gathering process. In case of limited storage, stale information at stationary nodes needs to be discarded when the storage capacity is approached. Hence, the quality of the information can also be measured by its completeness.

There are also types of WSNs where all nodes are mobile and need to cover the monitoring environment by autonomous moving [59]. In this case, tradeoffs between the energy cost of the system and its coverage can be explored.

6.2.2.2 Cooperation with Routing Diversity

Routing diversity has been considered in various studies [45, 66, 83, 50, 71]. Two models have been considered with different focus of routing technology. When multi-path routing is considered at the network layer, a simple disk model is often used to abstract the wireless transmission [45]. The key point is to identify a set of disjoint paths to route the packets such that a reliable packet delivery can be achieved above unreliable wireless communication. However, other than opportunistic data aggregation, existing techniques for multi-path routing have not formally addressed the issues for collaborative information processing with routing. To address these issues is challenging due to the fact that multiple information flows from the same source become available in the network. However, to efficiently use the advantage of data aggregation is crucial to reduce the relatively high cost of multi-path routing.

At the physical layer, routing diversity considers the exact behavior of fading channels using probabilistic models and exploits the broadcast property of wireless transmission to provide increased throughput and reliability [66, 83, 50, 71]. For example, the transmission of the same packet from multiple nodes can be coordinated to achieve the effect of multi-antenna transmission [66]. However, existing studies on routing diversity at physical layer also have not considered the collaboration between information processing and routing, which is an interesting and challenging topic in the future.

6.2.2.3 Integration with Sleep Scheduling

Sleep scheduling is also an important and widely used technology for low-traffic scenarios. It has been applied at various layers for different purposes, including reducing interference at physical layer, optimizing packet scheduling at MAC layer, enabling topology and routing control at network layer, and tuning sensing coverage at application layer. It is however not clear how to

integrate the techniques proposed in the thesis with existing techniques for sleep scheduling. In the following, we discuss several relevant issues.

From the perspective of the entire system, the topology and routing control enabled by sleep scheduling at network layer leads to re-construction of the network infrastructure from time to time. Hence, there should be certain mechanisms to signaling the change in the members of clusters, or the structure of the routing tree. Consequent adaptations to re-schedule the computation and communication tasks according to the changes are also necessary. For example, for the problem of information transportation over a given tree substrate, our on-line protocol can also be used to handle changes in the tree structure due to its feedback packet. However, for the problem of in-cluster processing, an on-line adaptation mechanism is missing. A basic approach is to re-calculate the task assignment every time change of the cluster occurs, hoping to amortize the cost of re-calculation in the case that such changes are rare. However, more efficient approaches are expected for real systems.

From the perspective of individual nodes, it is not clear how our rate adaptation based techniques can co-exist with sleep scheduling based protocols on a specific node. In fact, it is not clear how sleep scheduling policy designed at different layers could co-exist in the first place. It seems that coordination among these techniques are needed to provide an arbitration policy so that decisions for the most significant functionality can overwrite decisions for less critical needs. Considering the fact that there might be potentially multiple applications simultaneously running in the system, this kind of coordination is not always easy. For example, using rate adaptation on a sensor node may reduce both space and time re-usability for nodes in proximity, which may affect the behavior and performance of certain sleep scheduling based protocols.

Reference List

[1] Z. Abrams, A. Goel, and S. Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless senosr networks. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks*, Apr. 2004.

[2] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. on Information Theory*, 46:1204–1216, 2000.

[3] M. H. Ahmed, H. Yanikomeroglu, D. D. Falconer, and S. Mahmoud. Performance enhancement of joint adaptive modulation, coding and power control using cochannel-interferer assistance and channel reallocation. In *IEEE Wireless Communications and Networking Conference (WCNC)*, Mar. 2003.

[4] O. B. Akan and I. F. Akyildiz. ARC: The analytical rate control scheme for real-time traffic in wireless networks. *IEEE/ACM Trans. on Networking*, June 2004. to appear.

[5] S. Aldosari and J. Moura. Fusion in sensor networks with communication constraints. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks*, Apr. 2004.

[6] K. Arisha, M. Youssef, and M. Younis. Energy-aware TDMA-based MAC for sensor networks. In *IEEE Workshop on Integrated Management of Power Aware Communications, Computing, and Networking (IMPACCT)*, May 2002.

[7] E. Armanious, D. D. Falconer, and H. Yanikomeroglu. Adaptive modulation, adaptive coding, and power control for fixed cellular broadband wireless systems. In *IEEE Wireless Communications and Networking Conference (WCNC)*, Mar. 2003.

[8] ATMEL ATmega128L Datasheet. http://www.atmel.com.

[9] B. Awerbuch and Y. Azar. Buy-at-bulk network design. In *Annual Symposium on Foundations of Computer Science (FOCS)*, 1997.

[10] H. Aydin, R. Melhem, D. Mossé, and P. M. Alvarez. Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In *13th Euromicro Conference on Real-Time Systems*, June 2001.

[11] K. Balachandran, S. R. Kadaba, and S. Nanda. Channel quality estimation and rate adaption for cellular mobile radio. *IEEE J. of Selected Areas in Communication (JSAC)*, 17:1244–1256, 1999.

[12] K. Barr and K. Asanović. Energy aware lossless data compression. In *First International Conference on Mobile Systems, Applicatiosn and Services*, May 2003.

[13] G. Barriac, R. Mudumbai, and U. Madhow. Distributed beamforming for information transfer in sensor networks. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks*, Apr. 2004.

[14] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Annual Symposium on Foundations of Computer Science (FOCS)*, 1997.

[15] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Annual ACM Symposium on Theory of Computing (STOC)*, 1998.

[16] E. M. Belding-Royer, P. M. Melliar-Smith, and L. E. Moser. An analysis of the optimal node density for ad hoc mobile networks. In *IEEE International Conference on Communications (ICC)*, 2001.

[17] T. Bell, M. Powell, J. Horlor, and R. Arnold. The Canterbury Corpus. http://www.cosc.canterbury.ac.nz.

[18] P. Bergamo, S. Asgari, H. Wang, D. Maniezzo, L. Yip, R. E. Hudson, K. Yao, and D. Estrin. Collaborative sensor networking towards real-time acoustical beamforming in free-space and limited reverberance. *IEEE Trans. on Mobile Computing*, 3(3), July 2004.

[19] K. Bharat-Kumar and J. Jaffe. Routing to multiple destination in computer networks. *IEEE Trans. on Computers*, 3(31):343–351, 1983.

[20] R. Bhatia and M. Kodialam. On power efficient communication over multi-hop wireless netowks: Joint routing, scheduling and power control. In *IEEE InfoCom*, Mar. 2004.

[21] G. Boone. Reality mining: Browsing reality with sensor networks.

[22] D. Bradinsky and D. Estrin. Rumor routing algorithm for sensor networks. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Sep. 2002.

[23] T. D. Braun, S. Ali, H. J. Siegel, and A. A. Maciejewski. Using the Min-Min heuristic to map tasks onto heterogeneous high-performance computing systems. In *2nd Symposium of the Los Alamos Computer Science Institute*, Oct. 2001.

[24] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen. A dynamic voltage scaled microprocessor system. *IEEE J. of Solid-State Circuits*, 35(11), Nov. 2000.

[25] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen. Low-power CMOS digital design. *IEEE J. of Solid-State Circuits*, 27(4):473–484, Apr. 1992.

[26] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 85–96, July 2001.

[27] Compression ratios. http://www.cs.waikato.ac.nz/ singlis/ratios.html.

[28] M. Conard, M. Marrakchi, Y. Robert, and D. Trystram. Parallel Gaussian elimination on an MIMD computer. *Parallel Computing*, 6:275–295, 1988.

[29] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[30] Cougar Project. http://www.cs.cornell.edu/database/cougar.

[31] R. L. Cover and J. A. Thomas. *Elements of Information Theory*. John-Wiley and Sons Inc., 1991.

[32] R. Cristescu, B. Beferull-Lozano, and M. Vetterli. On network correlated data gathering. In *IEEE InfoCom*, Mar. 2004.

[33] R. L. Curz and A. V. Santhanam. Optimal routing, link scheduling and power control in multi-hop wireless networks. In *IEEE InfoCom*, Apr. 2003.

[34] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Sep. 2003.

[35] R. P. Dick, D. L. Rhodes, and W. Wolf. TGFF: Task graphs for free. In *International Workshop on Hardware/Software Codesign*, pages 97–101, Mar. 1998.

[36] T. ElBatt. On the scalability of hierarchical cooperation for dense sensor networks. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks*, Apr. 2003.

[37] T. ElBatt and A. Ephremides. Joint scheduling and power control for wireless ad hoc networks. *IEEE Trans. on Wireless Communications*, 3(1), 2004.

[38] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Symposium on Operating Systems Design and Implementation (OSDI)*, Dec. 2002.

[39] M. Enachescu, A. Goel, R. Govindan, and R. Motwani. Scale free aggregation in sensor networks. In *1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks (Algosensors)*, 2004.

[40] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 263–270, Aug. 1999.

[41] J. Fakcheroenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2003.

[42] G. Fohler and K. Ramamritham. Static scheduling of pipelined periodic tasks in distributed real-time systems. In *9th Euromicro Workshop on Real Time Systems*, 1997.

[43] A. E. Gamal, C. Nair, B. Prabhakar, E. Uysal-Biyikoglu, and S. Zahedi. Energy-efficient scheduling of packet transmissions over wireless networks. In *IEEE InfoCom*, June 2002.

[44] S. Ganeriwal, R. Kumar, S. Adlakha, and M. B. Srivastava. Network-wide time synchronization in sensor networks. Technical report, University of California, Los Angeles, 2002.

[45] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 251–254, July 2001.

[46] A. Giridhar and P. R. Kumar. Computing and communicating statistics in sensor networks. In *International Symposium on Information Theory*, June 2004.

[47] A. Goel and D. Estrin. Simultaneous optimization for concave costs: Single sink aggregation or single source buy-at-bulk. In *ACM-SIAM Symposium on Discrete Algorithms*, Jan. 2003.

[48] M. Goudreau, K. Lang, S. Rao, T. Suel, and T. Tsantilas. Towards efficiency and portability: programming with the bsp model. In *8th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 1–12, June 1996.

[49] F. Gruian and K. Kuchcinski. LEneS: Task scheduling for low-energy systems using variable supply voltage processors. In *Design Automation Conference (DAC)*, pages 449–455, 2001.

[50] M. Haenggi. Analysis and design of diversity schemes for ad hoc wireless networks. *IEEE J. of Selected Areas in Communication (JSAC)*, 23(1):19–27, Jan. 2005.

[51] S. Hanly and D. Tse. Power control and capacity of spread-spectrum wireless networks. *Automatica*, 35:1987–2012, 1999.

[52] W. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application specific protocol architecture for wireless microsensor networks. *IEEE Trans. on Wireless Communications*, 1(4):660–670, Oct. 2002.

[53] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Aug. 1999.

[54] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *9th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2000.

[55] G. Holland, N. Vaidya, and P. Bahl. A rate-adaptive MAC protocol for multi-hop wireless networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, July 2001.

[56] B. Hong and V. K. Prasanna. Optimizing a class of in-network processing applications in networked sensor systems. In *1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Oct. 2004.

[57] I. Hong, G. Qu, M. Potkonjak, and M. B. Srivastava. Synthesis techniques for low-power hard real-time systems on variable voltage processors. In *IEEE Real-Time Systems Symposium (RTSS)*, Dec. 1998.

[58] Y. T. Hou, Y. Shi, and J. Pan. A lifetime-aware flow routing algorithm for energy-constrained wireless sensor networks. In *IEEE MILCOM*, 2003.

[59] A. Howard, M. J. Mataricié, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *6th International Symposium on Distributed Autonomous Robotics Systems*, June 2002.

[60] C. Huang and R. D. Yates. Rate of convergence for minimum power assignment algorithm in cellular radio systems. *Wireless Networks*, 4:223–231, 1998.

[61] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A scalable and robust communication paradigm for sensor networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Aug. 2000.

[62] K. Kalpakis, K. Dasgupta, and P. Namjoshi. Maximum lifetime data gathering and aggregation in wireless sensor networks. In *IEEE International Conference on Networking (NETWORKS '02)*, pages 685–696, Aug. 2002.

[63] B. Kao and H. Garcia-Molina. Subtask deadline assignment for complex distributed soft real-time tasks. In *International Conference on Distributed Computing Systems (ICDCS)*, 1993.

[64] P. Karn. MACA: A new channel access method for packet radio. In *ARRL/CRRL Amateur Radio Computer Ntworking Conference*, Sep. 1990.

[65] B. Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Aug. 2000.

[66] A. E. Khandani, J. Abounadi, E. Modiano, and L. Zheng. Cooperative routing in wireless networks. In *Allerton Conference*, Oct. 2004.

[67] U. C. Kozat, I. Koutsopoulos, and L. Tassiulas. A framework for cross-lyaer design of energy-efficient communication with qos provisioning in multi-hop wireless networks. In *IEEE InfoCom*, Mar. 2004.

[68] B. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *International Workshop on Distributed Event-Based Systems*, 2002.

[69] B. Krishnamachari, Y. Mourtada, and S. Wicker. The energy-robustness tradeoff for routing in wireless sensor networks. In *IEEE International Conference on Communications (ICC)*, May 2003.

[70] M. Kubisch, H. Karl, A. Wolisz, L. C. Zhong, and J. Rabaey. Distributed algorithms for transmission power control in wireless sensor networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 16–20, Mar. 2003.

[71] J. N. Laneman, D. Tse, and G. W. Wornell. Cooperative diversity in wireless networks: Efficient protocols and outage behavior. to appear in IEEE Trans. on Information Theory.

[72] The LINDO System Inc. http://www.lindo.com.

[73] S. Lindsey, C. S. Raghavendra, and K. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Trans. on Parallel and Distributed Systems*, 13(9):924–935, Sep. 2002.

[74] G. Lu, B. Krishnamachari, and C. Raghavendra. An adaptive energy-efficient and low-latency mac for data gathering in sensor networks. In *International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks*, Apr. 2004.

[75] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel. Delay efficient sleep scheduling in wireless sensor networks. In *IEEE InfoCom*, Mar. 2005.

[76] J. Luo and N. K. Jha. Static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems. In *VLSI Design*, Jan. 2002.

[77] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a Tiny AGgregation service for ad-hoc sensor networks,. In *Symposium on Operating Systems Design and Implementation (OSDI)*, Dec. 2002.

[78] D. Marco, E. J. Duarte-Melo, M. Liu, and D. L. Neuhoff. On the many-to-one transport capacity of a dense wireless sensor network and the compressibility of its data. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks*, pages 1–16, Apr. 2003.

[79] G. P. McCormick. *Nonlinear Programming: Theory, Algorithms, and Applications.* John Wiley & Sons, 1982.

[80] P. Mejía-Alvarez, E. Levner, and D. Mossé. An integrated heuristic approach to power-aware real-time scheduling. In *Workshop on Power-Aware Computer Systems*, Feb. 2002.

[81] A. Meyerson, K. Munagala, and S. Plotkin. Cost-distance: Two metric network design. In *Annual Symposium on Foundations of Computer Science (FOCS)*, 2000.

[82] R. Min, M. Bhardwaj, S. Cho, A. Sinha, E. Shih, A. Wang, and A. P. Chandrakasan. Low-power wireless sensor networks. In *14th International Conference on VLSI Design*, pages 205–210, 2001.

[83] E. Moiano. Increasing reliability in ad hoc networks through diversity routing. In *International Workshop on Wireless Ad-Hoc Networks*, May 2004.

[84] R. A. Mucci. A comparison of efficient beamforming algorithms. *IEEE Trans. on Acoustic, Speech, Signal processing*, ASSP-22:548–558, June 1984.

[85] S. Narayanaswamy, V. kawadia, R. S. Sreenivas, and P. R. Kumar. Power control in ad-hoc networks: Theory, architecture, algorithm and implementation of the COMPOW protocol. In *European Wireless Conference - Next Generation WIreless Netwroks: Technologies, Protocols, Services and Applications*, Feb. 2002.

[86] PARSEC Project. http://pcl.cs.ucla.edu/projects/parsec.

[87] S. Pattem, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks*, Apr. 2004.

[88] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Comunications of the ACM*, 43(5):551–558, May 2000.

[89] B. Prabhakar, E. Uysal-Biyikoglu, and A. E. Gamal. Energy-efficient transmission over a wireless link via lazy packet scheduling. In *IEEE InfoCom*, Apr. 2001.

[90] J. Rabaey, J. Ammer, T. Karalar, S. Li, B. Otis, M. Sheets, and T. Tuan. PicoRadios for wireless sensor networks: The next challenge in ultra-low power design. In *International Solid-State Circuits Conference*, Feb. 2002.

[91] C. S. Raghavendra and S. Singh. PAMAS – power aware multi-access protocol with signaling for ad hoc networks. *ACM SIGCOMM Computer Communication Review*, 28(3):5–26, July 1998.

[92] V. Raghunathan, S. Ganeriwal, C. Schurgers, and M. B. Srivastava. E2WFQ: An energy efficient fair scheduling policy for wireless systems. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 30–35, Aug. 2002.

[93] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava. Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, March 2002.

[94] R. Ramanathan and Rosales-Hail. Topology control of multihop wireless networks using transmit power adjustment. In *IEEE InfoCom*, pages 404–413, Mar. 2000.

[95] V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. *IEEE J. of Selected Areas in Communication (JSAC)*, 8(17):1333–1344, 1999.

[96] V. Sarkar. *Partitioning and Scheduling Programs for Execution on Multiprocessors.* The MIT Press, Cambridge, Massachusetts, 1989.

[97] A. Scaglione and S. D. Servetto. On the interdependence of routing and data compression in multi-hop sensor networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Sep. 2002.

[98] C. Schurgers, O. Aberhorne, and M. B. Srivastava. Modulation scaling for energy-aware communication systems. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 96–99, Aug. 2001.

[99] C. Schurgers, V. Raghunathan, and M. B. Srivastava. Modulation scaling for real-time energy aware packet scheduling. In *IEEE GlobeCom*, Nov. 2001.

[100] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. B. Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE Trans. on Mobile Computing*, 1(1):70–80, Jan. 2002.

[101] K. Seada, M. Zuniga, B. Krishnamachari, and A. Helmy. Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In *ACM SenSys*, Nov. 2004.

[102] The Sensoria corporation. http://www.sensoria.com.

[103] S. Shakkottai, T. S. Rappaport, and P. C. Carlsson. Cross-layer design for wireless networks. *IEEE Wireless Communication Magazine*, pages 74–80, Oct. 2003.

[104] Y. Shin, K. Choi, and T. Sakurai. Power optimization of real-time embedded systems on variable speed processors. In *IEEE/ACM International Conference Computer-Aided Design*, pages 365–368, 2000.

[105] M. Singh and V. K. Prasanna. System level energy tradeoffs for collaborative computation in wireless networks. In *IEEE IMPACCT Workshop*, May 2002.

[106] M. Singh and V. K. Prasanna. A hierarchical model for distributed collaborative computation in wirelss sensor networks. In *5th Workshop on Advances in Parallel and Distributed Computational Models*, Apr. 2003.

[107] M. Singh and V. K. Prasanna. Supporting topographic queries in a class of networked sensor systems. In *Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS)*, Mar. 2005.

[108] A. Sinha and A. P. Chandrakasan. Dynamic power management in wireless sensor networks. *IEEE Design and Test of Cmputers*, 18(2):62–74, 2001.

[109] A. Sinha and A. P. Chandrakasan. Operating system and algorithmic techniques for energy scalable wireless sensor networks. In *2nd International Conference on Mobile Data Management*, Jan. 2001.

[110] A. Sinha, A. Wang, and A. Chandrakasan. Algorithmic transforms for efficient energy scalable computation. In *IEEE International Symposium on Low Power Electronics and Design*, Aug. 2000.

[111] D. Slepian and J. Wolf. Noiseless coding of correlated information sources. *IEEE Trans. on Information Theory*, IT-19(4):471–480, 1973.

[112] M. B. Srivastava, A. P. Chandrakasan, and R. W. Brodersen. Predictive system shutdown and other architectural techniques for energy efficient programmable computation. *IEEE Trans. on VLSI Systems*, 4(1):42–55, Mar. 1996.

[113] W. Su and I. F. Akyildiz. Time-diffusion synchronization protocol for sensor networks. Technical report, Georgia Institute of Technology, Broadband and Wireless Networking Laboratory, 2002.

[114] C. N. Taylor and S. Dey. Adaptive image compression for wireless multimedia communication. In *IEEE International Conference on Communications (ICC)*, June 2001.

[115] D. Tian and N. D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Sep. 2002.

[116] L. Tong, Q. Zhao, and G. Mergen. Multipacket reception in random access wireless networks: From singal procesisng to optimal medium access control. *IEEE Communications Magazine*, 39(11):108–112, Nov. 2001.

[117] T. Ue, S. Sampei, N. Morinaga, and K. Hamaguchi. Symbol rate and modulation level-controlled adaptive modulation/TDMA/TDD system for high-bit rate wireless data transmission. *IEEE Trans. on Vehicular Technology*, 47(4):1134–1147, Nov. 1998.

[118] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *ACM SenSys*, Nov. 2003.

[119] S. Verdu. *Multiuser Detection*. Cambridge Univ. Press, 1998.

[120] A. Wang, S.-H. Cho, C. G. Sodini, and A. P. Chandrakasan. Energy-efficient modulation and MAC for asymmetric microsensor systems. In *International Symposium on Low Power Electronics and Design (ISLPED)*, Aug. 2001.

[121] R. Wattenhofer, L. Li, B. Bahl, and Y. M. Wang. Distributed topoloty control for power efficient operation in multihop wireless ad hoc networks. In *IEEE InfoCom*, Apr. 2001.

[122] H. P. Williams. *Model Building in Mathematical Programming*. John Wiley & Sons, Ltd, 1999.

[123] The WINS Project, Rockwell Science Center. http://wins.rsc.rockwell.com.

[124] Y. Xu, J. Heidemann, and D. Estrin. Adaptive energy-conserving routing for multihop ad hoc networks. Technical Report 527, University of Southern California/Information Seicneces Institute, Oct. 2000.

[125] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, July 2001.

[126] T. Yang and A. Gerasoulis. DSC: Scheduling parallel tasks on an unbounded number of processors. *IEEE Trans. on Parallel and Distributed Systems*, 5(9):951–967, Sep. 1994.

[127] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 374–382, 1995.

[128] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *IEEE InfoCom*, June 2002.

[129] M. Younis, M. Youssef, and K. Arisha. Energy-aware routing in cluster-based sensor networks. In *International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Oct. 2002.

[130] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Issues in designing middleware for wireless sensor networks. *IEEE Network Magazine, special issue on Middleware Technologies for Future Communication Networks*, 18(1):15–21, Jan. 2004.

[131] Y. Yu and V. K. Prasanna. Energy-balanced multi-hop packet transmission in wireless sensor networks. In *IEEE GlobeCom*, Dec. 2003.

[132] Y. Yu and V. K. Prasanna. Energy-balanced task allocation for collaborative processing in wireless sensor networks. *ACM Mobile Networks and Applications (MONET)*, 10(1):115–131, 2005. special issue on Algorithmic Solutions for Wireless, Mobile, Ad Hoc and Sensor Networks.

[133] W. H. Yuen, H.-N. Lee, and T. D. Andersen. A simple and effective cross layer networking system for mobile ad hoc networks. In *PIMRC*, 2002.

[134] Y. Zhang and L. Cheng. Cross-layer optimization for sensor networks. In *New York Metro Area Networking Workshop*, Sep. 2003.

[135] Y. Zhang, X. Hu, and D. Z. Chen. Task scheduling and voltage selection for energy minimization. In *Design Automation Conference (DAC)*, 2002.

[136] L. C. Zhong, R. Shan, C. Guo, and J. Rabaey. An ultra-low power and distributed access protocol for broadband wireless sensor networks. In *IEEE Broadband Wireless Summit*, May 2001.

[137] C. Z. Zhou and B. Krishnamachari. Localized topology generation mechanisms for self-configuring sensor networks. In *IEEE GlobeCom*, Dec. 2003.

[138] D. Zhu, R. Melhem, and B. Childers. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multi-processor real-time systems. In *IEEE Real-Time Systems Symposium (RTSS)*, Dec. 2001.

Appendix A

Correctness of EMR-ALgo

Proof of Lemma 1: $\Rightarrow$ Condition (1) trivially holds for an optimal solution. Otherwise, we can always increase τ_i^* without violating the latency constraint and thus decrease the energy dissipation of V_i.

Let A denote a matrix of size $M \times v$, where $A[i][j] = 1$ if and only if $V_j \in p_i$. Intuitively, the 1's in the i-th row of A indicate the set of nodes on path p_i. Let $\vec{\Gamma}$ be a $M \times 1$ vector with all elements equal to Γ. Then OPTP can be expressed as to minimize $f(\vec{\tau})$, subject to $A\vec{\tau} = \vec{\Gamma}$. Based on the first-order necessary condition for linearly constrained problems [79], there exists a vector of values $\vec{\lambda} = (\lambda_1, \ldots, \lambda_M)^T$, such that

$$\nabla f(\vec{\tau^*}) + A^T \vec{\lambda} = 0 \tag{A.1}$$

where $\nabla f(\vec{\tau^*})$ is a column vector with the partial derivative $\frac{\partial f(\vec{\tau^*})}{\partial \tau_i^*} = \dot{w}_i(\tau_i^*)$ as the i-th element. By solving equation A.1, we have for any internal node, V_i,

$$\dot{w}_i(\tau_i^*) + \sum_{j:V_i \in p_j} \lambda_j = 0 \tag{A.2}$$

The sum in equation A.2 sums up λ_j's such that path p_j passes through V_i. Since any path containing the children of V_i must also pass through V_i, we have

$$\sum_{j:V_i \in p_j} \lambda_j = \sum_{(k,i)\in E} \{ \sum_{j:V_k \in p_j} \lambda_j \} \tag{A.3}$$

Thus, the necessary condition of optimality can be obtained by summing up equations A.2 for all the children of V_i and subtracting equation A.2 for V_i.

$\Leftarrow$ The proof follows the fact that the feasible space of OPTP is convex and compact. ■

Comments: We examine Lemma 1 in two interesting examples. In the first example, consider an aggregation tree $T = (V, E)$ where $V = \{V_1, V_2, V_3, V_4\}$ and $E = \{(1,3), (2,3), (3,4)\}$. Let $\vec{\tau^*}$ denote the optimal schedule of such a problem. Assume that $m_2 + m_3 \geq \Gamma$ and m_1 is fairly small compared with m_2 and m_3 such that we have $\tau_1^* = m_1$ in the optimal schedule. In this case, we shall have $\dot{w}_1(\tau_1^*) = 0$ and $\dot{w}_2(\tau_2^*) = \dot{w}_3(\tau_3^*)$, which respects Lemma 1.

In the second example, consider an OPTP problem with an extremely large latency constraint (e.g., $\Gamma = \infty$) such that the optimal schedule is obtained by setting $\tau_i^* = m_i$ for each $V_i \in V$. In such a case, we have $\dot{w}_i(\tau_i^*) = 0$, for each $V_i \in V$, which again respects Lemma 1.

More importantly, we shall note the following fact in the optimal solution for the second example. Let θ_i denote the start time for packet transmission from V_i. Ideally, θ_i can be determined as $\max_{(j,i)\in E}(\theta_j + \tau^* j)$. However, due to the laxity in the latency constraint, we can safely increase the start time of packet transmissions to some extent without compromising the optimality of the schedule. To prevent such situations, we adopt a strict definition of θ_i in the proof of Lemma 4 and Theorem 1 such that θ_i must equal $\max_{(j,i)\in E}(\theta_j + \tau^* j)$.

Corollary 2 *Consider an optimal schedule, $\vec{\tau^*}$, of OPTP; the following hold:*

1. *Suppose $\tau_i^* = m_i$ for some $V_i \in V$, we have $\tau_j^* = m_j$ for all sensor nodes in T_i.*

2. *Suppose $\tau_i^* < m_i$ for some $V_i \in V$, we have $\tau_j^* < m_j$ for all ancestors of V_i.*

Proof of Corollary 2:

1. Since $\tau_i^* = m_i$ implies $\sum_{(j,i)\in E} \dot{w}_j(\tau_j^*) = \dot{w}_i(\tau_i^*) = 0$, we have $\tau_j^* = m_j$ for all children of V_i. So on so forth, we have $\tau_j^* = m_j$ for all sensor nodes in the subtree rooted at V_i.

2. Let V_j denote the parent of V_i. $\tau_i^* = m_i$ implies $\dot{w}_j(\tau_j^*) \leq \dot{w}_i(\tau_i^*) < 0$; hence $\tau_j^* < m_j$. So on and so forth. ■

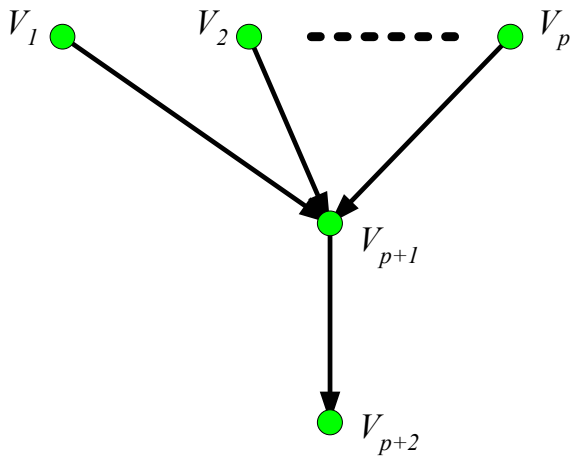


Figure A.1: A problem instance of 2-Lev-OPTP

We now define the level of a tree as the greatest number of edges contained by any path in the tree. We consider an OPTP problem with a two-level aggregation tree with exactly one internal node that has p children (see Figure A.1). We call the above problem 2-Lev-OPTP. Let V_{p+1} denote the internal node, with V_{p+2} denoting its parent and $\mathcal{C} = \{V_1, \ldots, V_p\}$ denote its children. Assume that for any $V_i \in \mathcal{C}$, a packet is ready to transmission at time θ_i and V_{p+2} must received aggregated information from V_{p+1} by time t. We first prove the following lemma:

Lemma 4 *Let $\vec{\tau^*} = \{\tau_1^*, \ldots, \tau_{p+1}^*\}$ denote an optimal schedule to the 2-Lev-OPTP problem as defined above, then the following hold:*

1. *The schedule $\vec{\tau^*}$ is unique.*

2. *Let θ_{p+1} denote the start time of packet transmission from V_{p+1} to V_{p+2} in the optimal schedule, i.e., $\theta_{p+1} = \max_{V_i \in \mathcal{C}}(\theta_i + \tau_i^*)$. Then θ_{p+1} never decreases when (a) some θ_i's, $V_i \in \mathcal{C}$, increase, holding t fixed; or (b) t increases, holding θ_i's fixed, for all $V_i \in \mathcal{C}$; or (c) both some θ_i's and t increase.*

   Let $\mathcal{D}$ denote the set of sensor nodes that increase their transmission start time in cases (a) and (c). Then, in particular, θ_{p+1} increases in case (a) if for any $V_i \in \mathcal{D}$, we have $\theta_{p+1} - \theta_i \leq m_i$; or in case (b) we have $t - \theta_{p+1} < m_{p+1}$; or in case (c) we have either of the previous two conditions hold.

3. *θ_{p+1} never increases when (a) some (≥ 1) θ_i's, $V_i \in \mathcal{C}$, decrease, holding t fixed; or (b) t decreases, holding θ_i's fixed, for all $V_i \in \mathcal{C}$; or (c) both some θ_i's and t decrease.*

   Let $\mathcal{D}'$ denote the set of sensor nodes that decrease their transmission start time in cases (a) and (c). Then, in particular, θ_{p+1} decreases in case (a) if for any $V_i \in \mathcal{D}'$, we have $\theta_{p+1} - \theta_i < m_i$; or in case (b) we have $t - \theta_{p+1} \leq m_{p+1}$; or in case (c) we have either of the previous two conditions hold.

Proof of Lemma 4:

1. The uniqueness of the optimal solution follows the strict convexity of the energy functions.

2. From Lemma 1, the optimal schedule $\vec{\tau^*}$ must satisfy $\sum_{i=1}^{p} \dot{w}_i(\tau_i^*) = \dot{w}_{p+1}(\tau_{p+1}^*)$. Moreover, we have $\tau_i^* \leq m_i$ for $i = 1, \ldots, p+1$. In the following, we prove property (a).

   We first assume that the transmission start time of exactly one child of V_{p+1} increases. That is, for some $V_a \in \mathcal{C}$, θ_a increases to θ_a'. Let $\hat{\theta}_{p+1}$ denote the start time of packet transmission from V_{p+1} in the resulting optimal schedule. We consider the following two cases.

   Case (i): Suppose that in schedule $\vec{\tau^*}$, $\tau_a^* \leq m_a$. This implies that $\theta_{p+1} = \theta_a + \tau_a^*$. Otherwise, a schedule with less energy dissipation than $\vec{\tau^*}$ can be constructed by increasing

τ_a^* by $\delta \leq \min\{\theta_{p+1} - (\theta_a + \tau_a^*), m_a - \tau_a^*\}$ without affecting τ_{p+1}^* or violating the latency constraint. Similarly, we have $\tau_{p+1}^* \leq m_{p+1}$ and $\theta_{p+1} + \tau_{p+1}^* = t$.

Now we consider the problem resulted from increasing θ_a to θ_a'. Suppose that in the new packet schedule, we still enforce the transmission by V_{p+1} to start at θ_{p+1}, we have:

$$\dot{w}_a(\tau_a^* - (\theta_a' - \theta_a)) + \sum_{V_i \in \mathcal{C} \wedge i \neq a} \dot{w}_i(\tau_i^*) < \sum_{V_i \in \mathcal{C}} \dot{w}_i(\tau_i^*) = \dot{w}_{p+1}(\tau_{p+1}^*) \; . \tag{A.4}$$

The above inequality comes from the fact that the first derivative of an energy function is strictly increasing due to its strict convexity.

Or, we may start the transmission by V_{p+1} at $\theta_{p+1} + (\theta_a' - \theta_a)$ in the new schedule. Since $\tau_{p+1}^* \leq m_{p+1}$, we have:

$$\begin{aligned} \dot{w}_a(\tau_a^*) + \sum_{V_i \in \mathcal{C} \wedge i \neq a} \dot{w}_i(\min\{\tau_i^* + (\theta_a' - \theta_a), m_i\} &\geq \sum_{V_i \in \mathcal{C}} \dot{w}_i(\tau_i^*) \\ &= \dot{w}_{p+1}(\tau_{p+1}^*) \\ &> \dot{w}_{p+1}(\tau_{p+1}^* - (\theta_a' - \theta_a)) \; . \end{aligned} \tag{A.5}$$

Equations A.4 and A.5 and the uniqueness of the optimal schedule imply that $\theta_{p+1} < \hat{\theta}_{p+1} < \theta_{p+1} + (\theta_a' - \theta_a)$.

Case (ii): Suppose that in schedule $\vec{\tau^*}$, $\tau_a^* = m_a$; hence, we have $\theta_{p+1} - \theta_a \geq m_a$. If $\theta_{p+1} - \theta_a' < m_a$, a similar analysis as in Case (i) can be carried out to show that $(\theta_{p+1} < \hat{\theta}_{p+1} < \theta_{p+1} + (\theta_a' - \theta_a)$. Otherwise, $\vec{\tau^*}$ remains an optimal schedule, implying that $\hat{\theta}_{p+1} = \theta_{p+1}$.

In case when multiple θ_i's ($V_i \in \mathcal{C}$) increase, it can handled by increasing these θ_i's one after another and the lemma still holds. Property (b) can be analyzed in a similar fashion. Also, Property (c) can be handled by first increasing θ_i's and then increasing t.

3. This part of the lemma is actually an inverse case of part (2) and can be easily proved by contradiction. ■

Now we present the proof of Theorem 1.

Proof of Theorem 1:

1. Recall that EMR-Algo works in iterations: for each iteration k, the algorithm determines s_i^k by decreasing i from $v-1$ to $M+1$. Since the EMR-Algo initializes $\theta_i = 0$ for $i = 1, \ldots, v-1$, it follows that $s_i^0 \leq s_i^1$ for each $i = 1, \ldots, v-1$. Suppose that $i' > 1$ and $k' > 1$ are the first time that there is a violation; that is, $s_{i'}^{k'} > s_{i'}^{k'+1}$.

   Consider the 2-level aggregation tree formed by $V_{i'}$ together with its parent, denoted as V_p, and its children, denoted as set $\mathcal{C}$. We have $s_p^{k'} \leq s_p^{k'+1}$, and $s_i^{k'-1} \leq s_i^{k'}$, for each $V_i \in \mathcal{C}$.

   From line 8 in EMR-Algo, the time stamps $s_p^{k'}$ and $s_i^{k'-1}$'s actually give the boundaries within which EMR-Algo determines $s_{i'}^{k'}$. Similarly, the time stamps $s_p^{k'+1}$ and $s_i^{k'}$'s give the boundaries within which EMR-Algo determines $s_{i'}^{k'+1}$. From part (2) of Lemma 4, we have $s_{i'}^{k'} \leq s_{i'}^{k'+1}$. This contradicts the assumption $s_{i'}^{k'} > s_{i'}^{k'+1}$ and hence property (1) holds.

2. It is obvious that $s_i^0 \leq s_i^*$, for each $i = 1, \ldots, v-1$. Similar to the proof for property (1), suppose that $i' \geq 1$ and $k' \geq 1$ are the first time that there is a violation; that is, $s_{i'}^{k'} > s_{i'}^*$.

   Again, consider the 2-level aggregation tree formed by $V_{i'}$ together with its parent, denoted as V_p, and its children, denoted as set $\mathcal{C}$. We have $s_p^{k'} \leq s_p^*$, and $s_i^{k'-1} \leq s_i^*$, for each $V_i \in \mathcal{C}$. The time stamps $s_p^{k'}$ and $s_i^{k'-1}$'s actually give the boundaries within which EMR-Algo determines $s_{i'}^{k'}$. Similarly, the time stamps s_p^* and s_i^*'s give the boundaries within

which EMR-Algo determines $s^*_{i'}$. Part (2) of Lemma 4 again leads to the contradiction that $s^{k'}_{i'} \leq s^*_{i'}$ and proves property (2).

3. We prove by contradiction and hence assume that $j = \max\{i : s^\infty_i < s^*_i\}$. Let V_p denote the parent of V_j and V_g denote the parent of V_p. We have $s^\infty_p = s^*_p$ and $s^\infty_g = s^*_g$. Since τ^*_j is optimal, we have $\tau^*_j \leq m_j$. We consider two cases:

   Case (i): We suppose that $\tau^*_j < m_j$. Considering the 2-level tree formed by V_p, V_g and the children of V_p, denoted as $\mathcal{C}$, we have $s^\infty_j < s^*_j$ and $s^\infty_i \leq s^*_i$, for each $V_i \in \mathcal{C} \wedge i \neq j$. Suppose that we run EMR-Algo for one more pass and let $\hat{\theta}_p$ denote the resulting start time for the transmission from V_p to V_g. From part (3) of Lemma 4, we have $\theta^\infty_p - (\theta^*_j - \theta^\infty_j) < \hat{\theta}_p < \theta^\infty_p = \theta^*_p$, contradicting both property (1) for V_p and the definition of j.

   Case (ii): We assume that $\tau^*_j = m_j$. From part (1) of Corollary 2, we have $\tau^*_i = m_i$ for any $V_i \in T_j$. Moreover, we have $\theta^\infty_p - \theta^\infty_j > \theta^*_p - \theta^*_j = \tau^*_j = m_j$. Obviously, we shall have $\tau^\infty_j = m_j$. Again from part (1) of Corollary 2, we have $\tau^\infty_i = m_i$ for any $V_i \in T_j$. Based on the definition of θ^∞_j and θ^*_j, we obtain the contradiction that $\theta^\infty_j = \theta^*_j$. ■