Information Processing and Routing in Wireless Sensor Networks

Yang Yu • Viktor K Prasanna • Bhaskar Krishnamachari

Information Processing and Routing in Wireless Sensor Networks

Information Processing and Routing in Wireless Sensor Networks

Yang Yu

Motorola Labs, USA

Viktor K Prasanna

Bhaskar Krishnamachari

University of Southern California, USA

World Scientific

NEW JERSEY · LONDON · SINGAPORE · BEIJING · SHANGHAI · HONG KONG · TAIPEI · CHENNAI

Published by

World Scientific Publishing Co. Pte. Ltd.
5 Toh Tuck Link, Singapore 596224
USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601
UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

INFORMATION PROCESSING AND ROUTING IN WIRELESS SENSOR NETWORKS

ISBN 981-270-146-X

Printed by Fulsland Offset Printing (S) Pte Ltd, Singapore

To our parents

Preface

Wireless sensor networks (WSNs) have become an important technology in the realization of 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. In the data-centric operating paradigm of WSNs, these two operations are tightly related and must be performed in a collaborative fashion.

A major concern in designing and operating WSNs 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. Cross-layer optimization in the context of collaborative information processing and routing is the motivation of this book.

Objectives:

This book presents state-of-the-art techniques for cross-layer optimization to improve the energy-efficiency of information processing and routing in wireless sensor networks. Besides providing a survey on this important research area, three specific research topics are addressed in detail: real-time information processing in a single hop cluster, real-time information transport over a given tree substrate, and information routing for computationally intensive applications. We choose these three topics because (1) each of them is important and challenging in itself, and (2) together they constitute a complete operating flow of information processing and routing. The presented techniques provide a framework above which various extensions can be overlaid.

We focus on the use of three system knobs for cross-layer optimization: voltage scaling, rate adaptation, and tunable compression. These system knobs can be used to explore the tradeoffs between communication/computation energy and latency, as well as the balance between computation and communication. The presented techniques are based on a high level system model that suitably abstracts details of low level hardware, networking protocols, and signal processing techniques.

We intend to (1) illustrate significant research results in cross-layer optimization for collaborative information processing and routing, and (2) motivate more research efforts in this crucial area from both theoretical and practical perspectives.

Book Organization:

In Chapter 1, we give an introduction to WSNs by discussing the enabling technologies of WSNs, the evolution of wireless sensor nodes, the application domain of WSNs, and related research topics and challenges.

In Chapter 2, we elaborate on the concept of information processing and routing in data-centric operating paradigm of WSNs. This is followed by a discussion of cross-layer optimization for energy minimization. A survey of state-of-the-art cross-layer optimization techniques is then presented.

The techniques presented in this book are centered around three system knobs: voltage scaling, rate adaptation, and tunable compression. In Chapter 3, we first give a list of common notations that are used throughout the book. We then describe the energy models of the above knobs. In particular, we demonstrate the tradeoffs involved in these knobs.

In Chapter 4, we consider **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 present integer linear programming formulations for optimal solutions as well as a 3-phase heuristic. Our techniques are shown to achieve up to 10x lifetime improvements in simulated scenarios.

In Chapter 5, we investigate 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.

In Chapter 6, we focus on 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 computationally 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 compression 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.

We expect that the contents presented in this book will motivate further research and implementation in the field of collaborative information processing and routing. Yet, many real-world challenges need to be overcome. We briefly discuss these challenges in Chapter 7.

Target Audience:

Most of the technical contents presented in this book are intended for scholars and researchers in academic institutions as well as industrial research groups. This book will benefit scholars and researchers by presenting (1) a survey of state-of-the-art techniques on information processing and routing and cross-layer optimization for energy-efficiency, and (2) a systematic framework to perform cross-layer optimization at various stages of information processing and routing, with results on three specific case studies.

This page is intentionally left blank

Contents

Chapter 1

Introduction to Wireless Sensor Networks

1.1 Overview

With the popularity of laptops, cell phones, PDAs, GPS devices, RFID, and intelligent electronics in the post-PC era, computing devices have become cheaper, more mobile, more distributed, and more pervasive in daily life. It is now possible to construct, from commercial off-the-shelf (COTS) components, a wallet size embedded system with the equivalent capability of a 90's PC. Such embedded systems can be supported with scaled down Windows or Linux operating systems. From this perspective, the emergence of wireless sensor networks (WSNs) is essentially the latest trend of Moore's Law toward the miniaturization and ubiquity of computing devices.

Typically, a wireless sensor node (or simply sensor node) consists of sensing, computing, communication, actuation, and power components. These components are integrated on a single or multiple boards, and packaged in a few cubic inches. With state-of-the-art, low-power circuit and networking technologies, a sensor node powered by 2 AA batteries can last for up to three years with a 1% low duty cycle working mode. A WSN usually consists of tens to thousands of such nodes that communicate through wireless channels for information sharing and cooperative processing. WSNs can be deployed on a global scale for environmental monitoring and habitat study, over a battle field for military surveillance and reconnaissance, in emergent environments for search and rescue, in factories for condition based maintenance, in buildings for infrastructure health monitoring, in homes to realize smart homes, or even in bodies for patient monitoring [60; 76; 124; 142].

After the initial deployment (typically ad hoc), sensor nodes are responsible for self-organizing an appropriate network infrastructure, often with multi-hop connections between sensor nodes. The onboard sensors then start collecting acoustic, seismic, infrared or magnetic information about the environment, using either continuous or event driven working modes. Location and positioning information can also be obtained through the global positioning system (GPS) or local positioning algorithms. This information can be gathered from across the network and appropriately processed to construct a global view of the monitoring phenomena or objects. The basic philosophy behind WSNs is that, while the capability of each individual sensor node is limited, the aggregate power of the entire network is sufficient for the required mission.

In a typical scenario, users can retrieve information of interest from a WSN 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. In this way, WSNs can be considered as a distributed database [45; 184]. It is also envisioned that sensor networks will ultimately be connected to the Internet, through which global information sharing becomes feasible (Figure 1.1).

The era of WSNs is highly anticipated in the near future. In September 1999, WSNs were identified by *Business Week* as one of the most important and impactive technologies for the 21st century [31]. Also, in January 2003, the *MIT's Technology Review* stated that WSNs are one of the top ten emerging technologies [125]. It is also estimated that WSNs generated less than $150 million in sales in 2004, but would top $7 billion by 2010 [133]. In December 2004, a WSN with more than 1000 nodes was launched in Florida by the ExScal team [61], which is the largest deployed WSN to date.

1.2 Enabling Technologies

1.2.1 *Hardware*

The hardware basis of WSNs is driven by advances in several technologies. First, System-on-Chip (SoC) technology is capable of integrating complete systems on a single chip. Commercial SoC based embedded processors from Atmel, Intel, and Texas Instruments have been used for sensor nodes such as UC Berkeley's motes [48; 173], UCLA's Medusa [120] and WINS [197], and MIT's μAMPS-1 [187]. Several research groups, such as the PicoRadio team from UC Berkeley [139], have been trying to integrate prototype sensor nodes (PicoNode I) onto a few chips (PicoNode II). Many interesting SoC

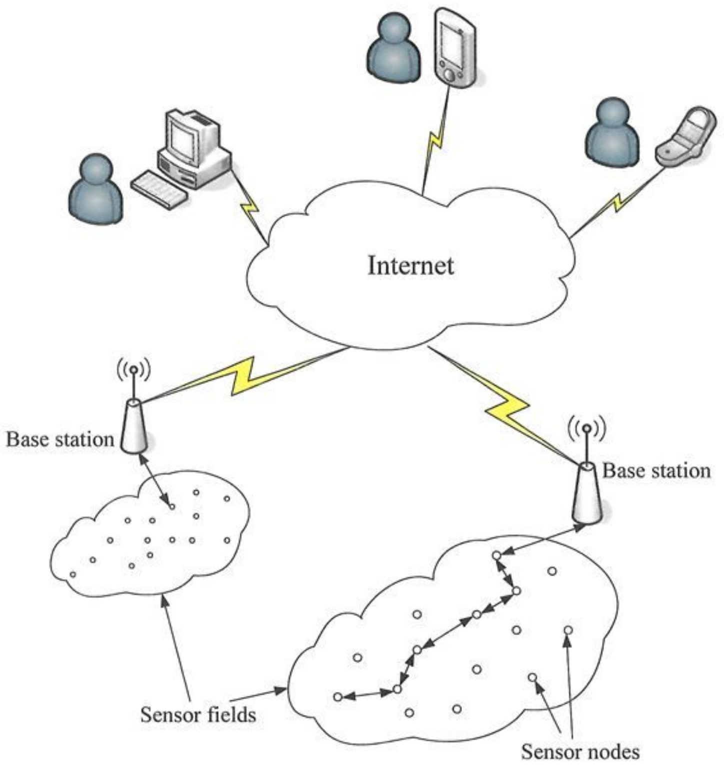


Fig. 1.1 Accessing WSNs through Internet.

designs related to wireless communication and sensor nodes can also be found at the SoC Design Challenge, 2004-2006 [174].

Second, commercial RF circuits enable short distance wireless communication with extremely low power consumption. Commercial products from RF Monolithics, Chipcon, Conexant Systems, and National Semiconductor have been used on various sensor nodes, including motes, Medusa, WINS, and μAMPS. A SoC based ZigBee radio is also available from Ember Cooperation [58]. These commercial radios can usually achieve a data rate of tens to hundreds of Kbps, while consuming less than 20 mW of power for both packet transmission and receiving [140]. With wideband technology, enhanced modulation schemes and error detection mechanisms are employed to provide increased robustness.

Third, Micro-Electro-Mechanical Systems (MEMS) technology [122] is now available to integrate a rich set of sensors onto the same CMOS chip. Commercially available sensors now include thermal, acoustic/ultrasound, and seismic sensors, magnetic and electromagnetic sensors, optical transducers, chemical and biological transducers, accelerometers, solar radiation detectors, photosynthetically active radiation detectors, and barometric pressure detectors [105]. These sensors can be used in a broad range of applications, including acoustic ranging, motion tracking, vibration detection, and environmental sensing.

The above technologies, along with advanced packaging techniques, have made it possible to integrate sensing, computing, communication, and power components into a miniaturized sensor node.

1.2.2 *Wireless Networking*

Besides hardware technologies, the development of WSNs also relies on wireless networking technologies. The 802.11 protocol, the first standard for wireless local area networks (WLANs), was introduced in 1997. It was upgraded to 802.11b with an increased data rate and CSMA/CA mechanisms for medium access control (MAC). Although designed for wireless LANs that usually consist of laptops and PDAs, the 802.11 protocols are also assumed by many early efforts on WSNs. However, the high power consumption and excessively high data rate of 802.11 protocols are not suitable for WSNs. This fact has motivated several research efforts to design energy efficient MAC protocols [109; 145; 189; 206]. Recently, the 802.15.4-based ZigBee protocol was released, which was specifically designed for short range and low data rate wireless personal area networks (WPAN). Its applicability to WSNs was soon supported by several commercial sensor node products, including MicaZ [48], Telos [140], and Ember products [58].

Above the physical and MAC layers, routing techniques in wireless networks are another important research direction for WSNs. Some early routing protocols in WSNs are actually existing routing protocols for wireless ad hoc networks or wireless mobile networks. These protocols, including DSR [88] and AODV [138], are hardly applicable to WSNs due to their high power consumption. They are also designed to support general routing requests in wireless networks, without considering specific communication patterns in WSNs. Nevertheless, the customization of these protocols for WSNs and the development of new routing techniques have become hot research topics [26; 51; 66; 73; 85; 95; 107; 160;

202]. The main idea behind these research efforts is to enable energy efficient and robust routing by exploiting link and path diversity.

1.2.3 *Collaborative Signal Processing*

Collaborative signal processing algorithms are another enabling technology for WSNs. While raw data from the environment are collected by sensor nodes, only useful information is of importance. Hence, raw data need to be properly processed locally at sensing nodes, and only processed data is sent back to the end users. Since computation is much more energy efficient than wireless communication, this avoids wasting energy on sending large volumes of raw data. Such signal processing is often required to be performed by a set of sensor nodes in proximity, due to the weak sensing and processing capabilities of each individual node.

Information fusion is an important topic for collaborative signal processing. Since sensor readings are usually imprecise due to strong variations of the monitoring entity or interference from the environment, information fusion can be used to process data from multiple sensors in order to filter noise measurements and provide more accurate interpretations of the information generated by a large number of sensor nodes. A rich set of techniques is applicable in this context, including Kalman filtering, Bayesian inference, neural networks, and fuzzy logic [7; 52; 91; 113; 165; 198].

Other signal processing techniques that have been developed for WSNs include time synchronization [57; 65; 179], localization [131; 154; 155], target tracking [50; 108; 214], edge and boundary detection [38; 101; 132], calibration [83; 194], adaptive sampling [137; 195], and distributed source coding [86; 153].

1.3 Evolution of Sensor Nodes

There has been a long history for (remote) sensing as a means for humans to observe the physical world. For example, the telescope invented in the 16th century is simply a device for viewing distant objects. As with many technologies, the development of sensor networks has been largely driven by defense applications.

1.3.1 *Military Networks of Sensors*

Since the early 1950s, a system of long-range acoustic sensors (hydrophones), called the Sound Surveillance System (SOSUS), has been deployed in the deep basins of the Atlantic and Pacific oceans for submarine surveillance. Beams from multiple hydrophone arrays are used to detect and locate underwater threats. Recently, SOSUS has been replaced by the more sophisticated Integrated Undersea Surveillance System.

Networks of air defense radars can be regarded as an example of networked large scale sensors. Both ground-based radar systems and Airborne Warning and Control System (AWACS) planes are integrated into such networks to provide all-weather surveillance, command, control, and communications. The radar dome on AWACS planes is 30 feet in diameter and six feet thick. It can detect flying targets in a range of more than 200 miles. In the 1980s and 1990s, the Cooperative Engagement Capability (CEC) [33] was developed as a military sensor network, in which information gathered by multiple radars was shared across the entire system, to provide a consistent view of the battle field.

Another early example of sensing with wireless devices is the Air Delivered Seismic Intrusion Detector (ADSID) system, used by US Air Force in the Vietnam war. Each ADSID node was about 48 inches in length, nine inches in diameter, and weighted 38 pounds. Equipped with a sensitive seismometer, these ADSID nodes were planted along the Ho Chi Minh Trail to detect vibrations from moving personnel and vehicles. The sensed data were transmitted from each node directly to an airplane, over a channel with unique frequency.

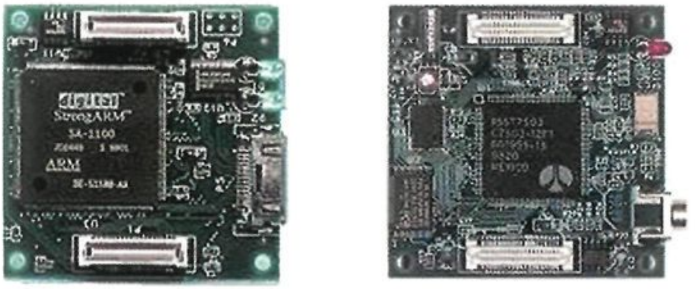
Although the ADSID nodes were large, and the high energy cost of direct communication limited the lifetime of nodes to only a few weeks, they successfully demonstrated the concept of wirelessly networked sensors. With the success of digital packet radios for wireless networking by the ALOHAnet Project [2] at Hawaii and DARPA's Packet Radio Project [90] in 1970s, wireless communication within the same frequency band using MAC techniques and packet-based multihop communication became possible.

1.3.2 *Next Generation Wireless Sensor Nodes*

1.3.2.1 *WINS from UCLA*

In 1996, the Low Power Wireless Integrated Microsensors (LWIMs) [28] were produced by UCLA and the Rockwell Science Center. By using commercial, low cost CMOS fabrication, LWIMs demonstrated the ability to integrate multiple sensors, electronic interfaces, control, and communication on a single device. LWIM supported over 100 Kbps wireless communication at a range of 10 meters using a 1 mW transmitter.

In 1998, The same team built a second generation sensor node — the Wireless Integrated Network Sensors (WINS) [11]. Commercial WINS from Rockwell Science Center [197] each consists of a processor board with an Intel StrongARM SA1100 32-bit embedded processor (1 MB SRAM and 4 MB flash memory), a radio board that supports 100 Kbps with adjustable power consumption from 1 to 100 mW, a power supply board, and a sensor board. These boards are packaged in a 3.5"x3.5"x3" enclosure (Figure 1.2). The processor consumes 200 mW in the active state and 0.8 mW when sleeping.

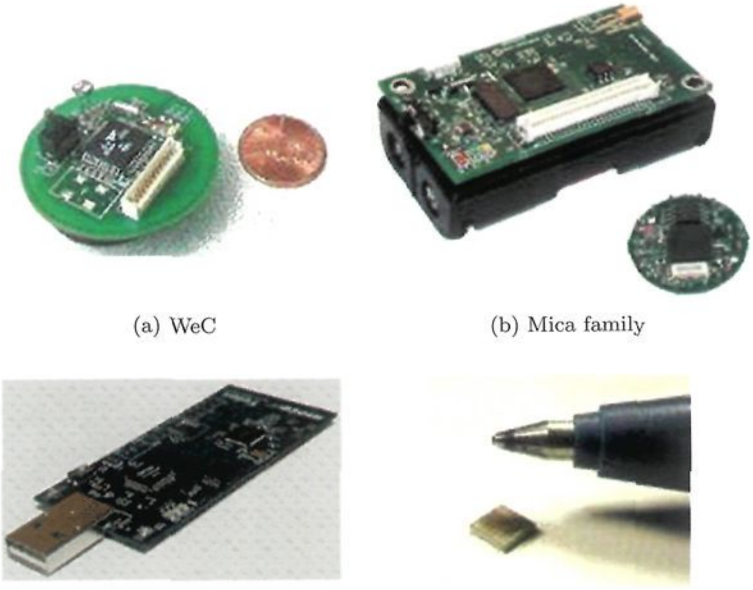


(a) The WINS processor board (b) The WINS radio board

Fig. 1.2 WINS node from Rockwell Science Center.

1.3.2.2 *Motes from UC Berkeley*

While WINS offer relatively powerful processing and communication capabilities, other research efforts have been developing smaller and cheaper nodes with less power consumption. In 1999, the Smart Dust project [173] at UC Berkeley released the first node, WeC, in their product family of *motes* (Figure 1.3(a)). WeC was built with a small 8-bit, 4 MHz Atmel microcontroller (512 bytes RAM and 8 KB flash memory), which consumed 15 mW active power and 45 μW sleeping power. WeC also had a simple radio supporting a data rate up to 10 Kbps, with 36 mW transmitting power and 9 mW receiving power. Later on, René and Dot were built in 1999 and 2000, respectively, with upgraded microcontrollers.


(a) WeC (b) Mica family

(c) Telos (d) Spec prototype

Fig. 1.3 Motes from UC Berkeley.

Along this line, the Mica family was released in 2001, including Mica [75], Mica2, Mica2Dot, and MicaZ [48]. While Mica still used an 8-bit 4 MHz microcontroller (ATmega103L), it offered enhanced capabilities in terms of memory and radio, compared with preceding products. Specifically, Mica was designed with 4 KB Ram, 128 KB flash, and a simple bit-level radio using RFM TR1000 that supported up to 40 Kbps with almost the same power consumption as the radio module on WeC. Mote architecture allowed several different sensor boards, or a data acquisition board, or a network interface board to be stacked on top of the main processor/radio board. These boards supported various sensors, most of which are listed in Section 1.2.1. The basic processor/radio board was approximately *one inch by two inches in size* (Figure 1.3(b)).

The follow-ups to Mica, Mica2 and Mica2Dot were built in 2002 with an ATmega128L microcontroller that reduced standby current (33 mW active power and 75 μW sleep power). They also had improved radio modules (Chipcon CC1000) with more options for frequency range, and increased resilience to noise by using FSK modulation. One year later, MicaZ was produced with a Chipcon CC2420 wideband radio module that supported 802.15.4 and ZigBee protocols, with a data rate up to 250 Kbps. This radio module also supported on-chip data encryption and authentication.

The latest member in the family, Telos [140], was released in 2004 (Figure 1.3(c)). Telos offered a set of new features: (1) a microcontroller from Texas Instruments with 3 mW active power and 15 μW sleep power, (2) an internal antenna built into the printed circuit board to reduce cost, (3) an on-board USB for easier interface with PCs, (4) integrated humidity, temperature, and light sensors, and (5) a 64-bit MAC address for unique node identification.

An interesting research testbed is the Spec platform [74], which integrated the functionality of Mica onto a single 5 mm^2 chip (Figure 1.3(d)). Spec was built with a micro-radio, an analog-to-digital converter, and a temperature sensor on a single chip, which lead to a 30-fold reduction in total power consumption. This single-chip integration also opened the path to low cost sensor nodes.

The integrated RAM and flash memory architecture has greatly simplified the design of the mote family. However, the tiny footprint also requires a specialized operating system, which was developed by UC Berkeley, called TinyOS [185]. TinyOS features a component-based architecture and event-driven model that are suitable for programming with small embedded devices, such as motes. The combination of Motes and TinyOS is gradually becoming a popular experimental platform for many research efforts in the field of WSNs.

1.3.2.3 *Medusa from UCLA*

The design philosophy and operational space of motes are quite different from those of WINS. On one hand, motes are designed for simple sensing and signal processing applications, where the demand for computation and communication capabilities is low. On the other hand, WINS are essentially an embedded version of PDAs, for more advanced computationally intensive applications with large memory space requirements. To bridge the gap between the two extremes, the Medusa MK-2 sensor node was developed by the Center for Embedded Networked Sensing (CENS) at UCLA in 2002 (Figure 1.4).

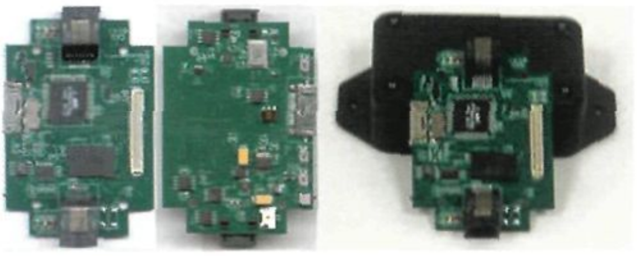


Fig. 1.4 Medusa node from UCLA.

One distinguishing feature of Medusa MK-2 is that it integrates two microcontrollers. The first one, ATmega128, is dedicated to less computationally demanding tasks, including radio base band processing and sensor sampling. The second one, AT91FR4081, is a more powerful microcontroller (40 MHz, 1 MB flash, 136 KB RAM) that can be used to handle more sophisticated, but less frequent signal processing tasks (e.g., the Kalman filter). The combination of these two microcontrollers provides more flexibility in WSN development and deployment, especially for applications that require both high computation capabilities and long lifetime.

1.3.2.4 *PicoRadio from UC Berkeley*

All the aforementioned sensor architectures are based on batteries. Due to the slow advancement in battery capacity, techniques for energy scavenging from the environment have been an attractive research field. In 2003, the Berkeley Wireless Research Center (BWRC) presented the first radio transmitter, PicoBeacon (Figure 1.5), purely powered by solar and vibrational energy sources. With a custom RF integrated circuitry that was developed for power consumption less than 400 μW, the beacon was able to achieve duty cycles up to 100% for high light conditions and 2.6% for typical ambient vibrational conditions. It is anticipated that an integrated wireless transceiver with < 100 μW power consumption is feasible in the near future.

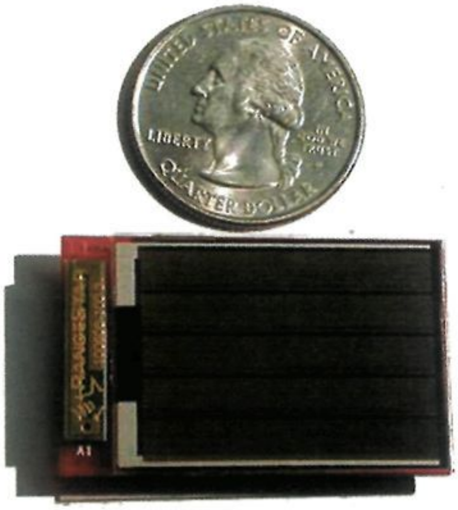


Fig. 1.5 PicoBeacon from UC Berkeley.

The BWRC also produced SoC based sensor nodes instead of using COTS components. In 2002, PicoNode II was built using two ASIC chips that implemented the entire digital portion of the protocol stack. Together, the chip set consumed an average of 13 mW when three nodes were connected. The team is also building PicoNode III, which will integrate a complete PicoNode into a single small aspect-ratio package.

1.3.2.5 *μAMPS from MIT*

The same ASIC based approach is being taken by the μAMPS group from MIT. Following its first testbed, μAMPS-I (Figure 1.6), the team is now trying to build a highly integrated sensor node comprised of a digital and an analog/RF ASIC, μAMPS-II. The interesting feature of μAMPS-II is that the node will be able to operate in several modes. It can operate as either a low-end stand-alone guarding node, a fully functional node for middle-end sensor networks, or a companion component in a more powerful high-end sensor systems. Thus, it favors a network with heterogeneous sensor nodes for a more efficient utilization of resources.

Besides the above sensor nodes, other commercial products and testbeds for WSNs include Ember products [58], Sensoria WINS [161], Pluto mote [40], PC104 testbed [136], and Gnome testbed [193].

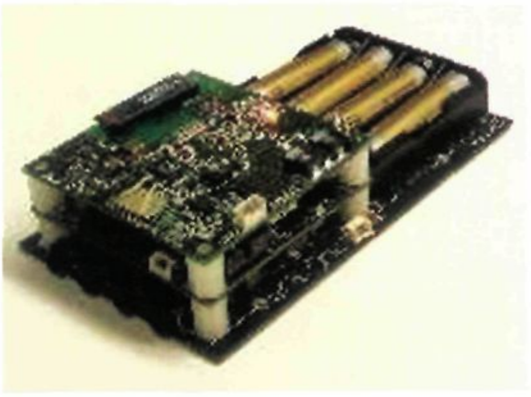


Fig. 1.6 μAMPS-I from MIT.

1.3.3 *Why Microscopic Sensor Nodes?*

The transition from large to small scale sensor nodes has several advantages.

(1) Small sensor nodes are easy to manufacture with much lower cost than large scale sensors. They are even disposable if the envisioned US$1 target price can be realized in the future.
(2) With a mass volume of such low cost and tiny sensor nodes, they can be deployed very closely to the target phenomena or sensing field at an extremely high density. Therefore, the shorter sensing range and lower sensing accuracy of each individual node are compensated for by the shorter sensing distance and large number of sensors around the target objects, which generates a high signal to noise ratio (SNR).
(3) Since computing and communication devices can be integrated with sensors, large-sample in-network and intelligent information fusion becomes feasible. The intelligence of sensor nodes and the availability of multiple onboard sensors also enhances the flexibility of the entire system.
(4) Due to their small size and self-contained power supply, sensor nodes can be easily deployed into regions where replenishing energy is not available, including hostile or dangerous environments. The survivability of nodes also increases with reduced size.

(5) The high node density enables system-level fault tolerance through node redundancy.

These advantages are illustrated by the microclimate monitoring of coastal redwood trees [150]. It is known that the movement of water from the ground to the canopy through the trunk is caused by the difference in water vapor pressure in the leaf and water vapor pressure in the air. To understand precisely the effects of microclimate variables, such as temperature and humidity, it is necessary to gather such information at different locations on the tree.

Because of their coarse resolution, it is difficult for large scale sensors, such as weather stations, to perform this task. However, by mounting a sufficient number of small sensor nodes along the tree trunk, it is possible to gather the desired information with a relatively low cost. These sensor nodes are able to collect both spatially and temporally dense sampling to enable a comprehensive view of the microclimate around the redwood tree. Because of wireless networking, it is easy to add more sensor nodes or move mounted nodes for better coverage. It is also possible to place redundant sensor nodes in order to enable local information fusion for better sensing accuracy.

Once deployed, the long lifetime of the network allows data collection over several years. The in-network storage capacity makes it possible to transfer intermittently the gathered data to a laptop. Also, these autonomous and intelligent sensor nodes are able to self-organize and self-heal the wireless network should node or link failures occur. This untethered operation avoids costly human management and maintenance.

1.4 Applications of Interest

An outline of the envisioned applications for WSNs is given in [11]. Descriptions of general applications for WSNs can also be found in [39] and [199]. For the purpose of this book, we categorize the applications into two classes. The first class, data gathering applications, focuses on entity monitoring with limited signal processing requirements. The primary goal of these applications is to gather information of a relatively simple form, such as temperature and humidity, from the operating environment. Some environmental monitoring and habitat study applications also belong to this class.

The second class of applications require the processing and transportation of large volumes of complex data. This class includes heavy industrial monitoring and video surveillance, where complicated signal processing algorithms are usually employed. We refer to these applications as computationally intensive applications.

In the following sections, we describe several academic and industrial efforts based on the above categorization. While both classes of applications are important for realizing the potential of WSNs, the involved techniques can be quite different due to their varying computation and communication demands. In Section 1.6, we discuss these differences in the context of this book.

1.4.1 *Data Gathering Applications*

1.4.1.1 *Habitat Study*

Habitat study is one of the driving applications for WSNs [34]. Such applications usually require the sensing and gathering of bio-physical or biochemical information from the entities under study, such as Redwoods [150], Storm Petrels [116], Zebras [89], and Oysters [84]. In many scenarios, habitat study requires relatively simple signal processing, such as data aggregation using minimum, maximum, or average operations. Hence, motes are ideal platforms for such applications.

The famous Great Duck Island project was initiated in the Spring of 2002 by Intel Research and UC Berkeley, to monitor the microclimates in and around Storm Petrel nesting burrows [116]. Thirty two motes were deployed on the island, each equipped with sensors for temperature, humidity, barometric pressure, and mid-range infrared. The network was designed to have a tiered structure. The motes were grouped into patches so that data collected in each patch could be relayed via a gateway to a base station, where data logging was performed. Within one year of monitoring, the system gathered approximately 1 million readings. In 2003, a second generation network, with more than 100 nodes, was also deployed.

Cape Breton University and the National Research Council of Canada are conducting an on-going bio-physical monitoring effort in the bras d'Or Lakes. Their goal is to study the life cycle of an oyster parasite (MSX), requiring the gathering of temperature and salinity parameters [84]. COTS sensor nodes will be deployed in the shallow shoreline of the lakes, which is preferred by oysters and easily accessible for biological and oceanographic monitoring.

1.4.1.2 *Environmental Monitoring*

Environmental monitoring is another application for WSNs. The vast spaces involved in such applications require large volumes of low cost sensor nodes that can be easily dispersed throughout the region. For instance, WSNs have been studied for forest fire alarm [99], landscape flooding alarm [8], soil moisture monitoring [32], microclimate and solar radiation mapping [141], and environmental observation and forecasting in rivers [43].

Researchers at University of West Australia are developing a prototype WSN for outdoor, fine-grained environmental monitoring of soil water [32]. Such a network can be used to assist salinity management strategies, or to monitor irrigated crops, urban irrigation, and water movement in forest soils. In January 2005, a prototype network was built, which included 15 Mica2 nodes integrated with soil moisture sensors and other gateway and routing nodes. The system distinguishes itself by using a reactive data gathering strategy — frequent soil moisture readings are collected during rain, while less frequent readings are collected otherwise. This strategy helps increase the system lifetime.

1.4.2 *Computation-Intensive Applications*

1.4.2.1 *Structural Health Monitoring*

Health monitoring for civil structures has long been a research topic for industry and academia. Traditional methods include visual inspection, acoustic emission, ultrasonic testing, and radar tomography. The emergence of WSNs has prompted new, non-destructive, and cheap methods for many tasks related to structural health monitoring [114; 178; 200].

The volume of raw data to be gathered and transported for such applications is on the order of 1-10 Mbps [37]. Thus, transmitting only useful information obtained from local signal processing becomes imperative for sustaining a long system lifetime. Many sophisticated and computationally intensive signal processing algorithms have been studied, including the Fast Fourier Transformation (FFT), Wavelet Transform, Autoregressive Models [175], and AR-ARX Damage Detection Pattern Recognition Method [175]. To serve the large computation demand from these algorithms, while maximizing energy savings, a dual-core design method has been employed. For instance, with the aforementioned Medusa node and the sensor node developed by Lynch [114], while a low-end microcontroller is responsible for frequent sensing and communication tasks, a high-end embedded processor is occasionly utilized when heavy signal processing is required.

An on-going Structural Health Monitoring (SHM) project by University of Southern California [164] has developed two software systems, Wisden and NET-SHM. These systems facilitate continuous data acquisition over a self-configuring multi-hop WSN, with high data rate and reliable communication requirements. Moreover, a full-scale testbed ceiling of 28×48 feet has been built with actuators to deliver deterministic excitations. Currently, the team is constructing robotic actuators that can be remotely controlled to move above the ceiling. The team is also investigating the use of other modalities, such as images, to enhance the fidelity of the system.

1.4.2.2 *Heavy Industrial Monitoring*

Sensors have already been widely used in industrial applications, such as the monitoring of automated assembly lines. Integrating wireless technology with these sensors enables condition based maintenance (CBM) to reduce downtime and enhance safety, with low installation and maintenance cost. Condition based maintenance can replace traditional high-cost, schedule-driven, manual maintenance for various industrial entities, including power plants, oil pipelines, transportation systems and vehicles, engineering facilities, and industrial equipment.

Industrial applications are unique in their requirement of highly reliable operation in harsh environments. For example, the electromagnetic radiation of machines may cause microcontroller malfunction or wireless communication interference. Also, the large variation in temperature and humidity demands reliable hardware components. Moreover, industrial applications often require the processing of large volumes of data with sophisticated signal processing algorithms. Thus, computation demand is usually high for these applications.

Intel Research has deployed a network with 160 Mica2 motes on a ship to measure the vibrations in the ship's pumps, compressors, and engines as an indicator of potential failure [29; 54]. These motes were organized into clusters, with Stargate gateways [48] forming the backbone of the network. Without operator intervention, the deployed network operated for 4 months without major failures. This experiment was still preliminary since the diagnosis of the ship equipments was performed in a centralized way at the base station, instead of distributed within the network. However, it paved the path for WSNs to a broad range of applications in industrial environments.

1.5 Research Topics and Challenges

Due to potentially harsh, uncertain, and dynamic environments, WSNs are envisioned to operate in an autonomous and untethered fashion. This poses considerable challenges ranging through network organization, topology discovery, communication scheduling, routing control, and signal processing. Also, tight energy budgets enforce energy efficient designs for hardware components, network stacks, and application algorithms.

In this section, we briefly describe a list of research challenges for WSNs. For the purpose of this book, we are particularly interested in the first three challenges. In Chapter 2, we discuss them in detail.

(1) **Data-centric paradigm**: 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, the data-centric paradigm is unique in several ways. New communication patterns resemble a reversed multicast tree. In-network processing extracts information from raw data and removes redundancy among multiple source data. Also, cooperative strategies among sensor nodes are used to replace the non-cooperative strategies for most Internet applications. The development of appropriate routing strategies that take the above factors into consideration is challenging.

(2) **Collaborative information processing and routing**: The data-centric paradigm involves two fundamental operations in WSNs: information processing and information routing. Many research efforts are motivated by the fact that information processing and routing are mutually beneficial. While information processing helps reduce the 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 in conjunction with joint compression for energy minimization turns out to be NP-hard.

(3) **Energy-efficient design**: Once deployed, it is often infeasible or undesirable to re-charge sensor nodes or replace their batteries. Thus, energy conservation becomes crucial for sustaining a sufficiently long network lifetime. Among the various techniques proposed for improving energy-efficiency, cross-layer optimization has been realized as an effective approach. Due to the nature of wireless communication, one performance metric of the network can be affected by various factors across 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.

(4) **Network discovery and organization**: Due to the large scale of WSNs, each sensor node behaves based on its local view of the entire network, including topology and resource distribution. Here, resources include battery energy and sensing, computation, and communication capabilities. To establish such a local view, techniques such as localization and time synchronization are often involved. A local view depends on the initial deployment of sensor nodes, which is itself a challenging topic. The network is usually organized using either a flat or hierarchical structure, above which topology control, MAC, and routing protocols can be applied accordingly.

    One key challenge is to handle network dynamics during the process of network discovery and organization. These dynamics include fluctuation in channel quality, failure of sensor nodes, variations in sensor node capabilities, and mobility or diffusion of the monitored entity. Autonomous adaptation of network discovery and organization protocols, in light of such dynamics, is the key to deliver proper system functionality.

(5) **Security**: Since WSNs may operate in a hostile environment, security is crucial to ensure the integrity and confidentiality of sensitive information. To do so, the network needs to be well protected from intrusion and spoofing. The constrained computation and communication capability of sensor nodes make it unsuitable to use conventional encryption techniques. Lightweight and application-specific architectures are preferred instead.

1.6 Focus of This Book

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

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

(1) information processing within a cluster of sensor nodes (or in-cluster information processing)
(2) information transportation over a given multi-hop tree structure (referred to as data gathering tree)
(3) information routing for computationally intensive applications over a general graph.

Each of these three topics is important and challenging in itself. Together, they cover a complete operating flow, from raw data sensing and processing at local clusters to information gathering and routing across the network. This is the major motivation to choose these three topics.

To facilitate cross-layer optimization in these topics, we study a set of fundamental techniques, referred to as *system knobs*. These system knobs are parameters that are exposed at certain levels, and can be tuned to adjust the performance of the system. In this book, we are particularly interested in three of them: voltage scaling, rate adaptation, and tunable compression. These techniques address the energy issue 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 cost. We illustrate these tradeoffs in Figure 1.7.

These three system knobs are applied in the aforementioned research topics. For the first topic, we investigate the application of voltage scaling and rate adaptation to maximize the system lifetime for in-cluster processing. For the second topic, we study rate adaptation for minimizing the energy cost for information transporting over an existing tree. For the last topic, we show that tunable compression can be incorporated into routing tree construction for minimizing the overall computation and communication energy in information routing.

One scenario for our research efforts is the cluster-based network scheme [72; 167; 208; 207]. In this scheme, the whole network is parti-

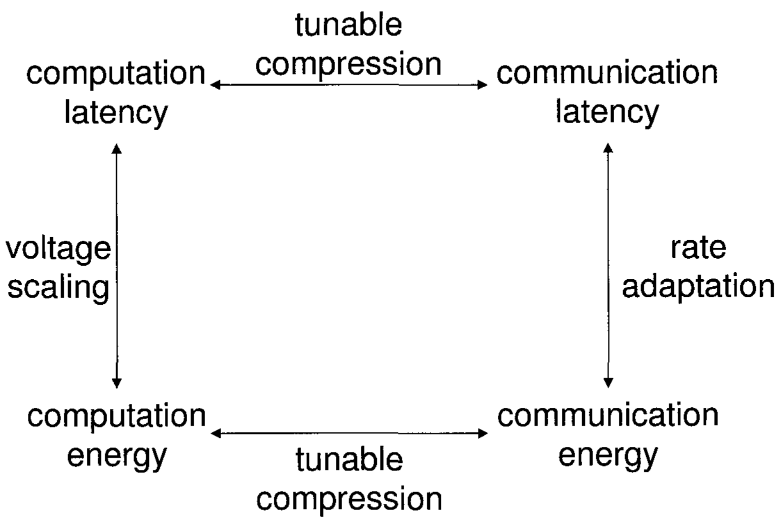


Fig. 1.7 Tradeoffs explored by three system knobs: voltage scaling, rate adaptation, and tunable compression.

tioned into either static or dynamic clusters, with one sensor node per 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 [72], a multi-hop tree that consists of only cluster heads [167], or a general multi-hop tree consisting of any sensor nodes in the network [208]. While the construction of a cluster-based infrastructure is beyond the scope of this book, we can see that our three research topics fit well into this scheme. Moreover, the proposed techniques are applicable to other scenarios as well.

Note that the research efforts presented in the book by no means provide a complete solution to information processing and routing. Our works are based on a relatively high model of the system. We are not concerned with the details of specific hardware to realize the system knobs, 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 a cross-layer optimization perspective, our work sits between the hardware and application layers when voltage scaling is employed, MAC and application layers for rate adaptation, and routing and application layers for tunable compression.

This page is intentionally left blank

Chapter 2

Background

2.1 Data-Centric Paradigm

The set of applications envisioned for WSNs include environment monitoring, habitat study, battlefield surveillance, and infrastructure monitoring. All of these applications require the sensing, processing, and gathering of information from the physical environment, in which the network operates. The end user is interested in the content of the information (including related spatial and temporal specifications), instead of the sensor nodes that sense or hold the information. This focus on information is known as *data-centric* computation and communication [100] (as opposed to the address-centric or node-centric paradigm, which are exhibited in traditional Internet networks, mobile ad hoc networks, and parallel and distributed systems).

The data-centric paradigm differs from the address-centric paradigm in several ways. First, the typical goal of data-centric WSNs is to gather specific information from multiples source nodes, and transfer it to a small number of sink nodes. Thus, the communication pattern normally resembles *a reversed-multicast tree*, instead of a more randomized peer-to-peer-based communication pattern found in address-centric systems. The work presented in this book is largely motivated by this tree structure.

Second, raw data gathered from a physical environment are usually not of direct interest to the end user, and are often strongly correlated. Therefore, it is necessary to process the data before they are transported to the end user. Such *in-network processing* includes signal/image processing and data fusion to extract useful information from the raw data, data compression to reduce communication load, and joint compression of multiple sources to eliminate redundancy. Our work addresses these three forms of in-network processing.

Third, the routing schemes for address-centric systems are based on fulfilling each individual data routing request. Thus, they become unsuitable for data-centric systems, where the communication request of the whole system is application-specific, e.g., routing 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 than address-centric applications, 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 resources of the system can be more efficiently utilized.

The difference between data-centric and address-centric paradigms can be effectively reflected by the user queries. For example, typical user queries or service requests in the address-centric paradigm are "Load the web page at address ...," "Transfer file A from host B to host C," etc. However, in a data-centric system, typical queries are "Gather the average temperature at region A within time T," or "Count the number of vehicles currently in region A." For these queries, the user is interested purely in the information itself, but not the sensor nodes that generate or hold the information, nor the way the information is transmitted to the end user.

The unique features of the data-centric paradigm have raised various new research topics for WSNs, ranging from hardware components, to networking techniques, to application algorithms. One of the most important topics is the method of processing and transporting information within the system so that the required functionality and performance metrics can be fulfilled. We discuss this topic in detail in the next section.

2.2 Collaborative Information Processing and Routing

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

(1) *information processing*, which includes sensing the environment and extracting useful information from the raw data, and
(2) *information routing*, which includes combining the information from different sources across the network, and routing the final set of information to the end user.

Note that in some sense these two operations are applicable to the 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 a data-centric system, these two operations are not sequential and independent, as they usually are in the address-centric paradigm. In the data-centric system, these two operations are performed in a parallel fashion and are closely related to each other.

Information processing can help reduce the data volume to route by information extraction from the raw data, data compression, or redundancy elimination through joint source compression. Also, since source data are distributed across the network, the routing scheme determines the available data for many information processing techniques (especially joint source compression). Such a mutually beneficial relationship leads to a tightly coupled design and implementation of information processing and routing. In this book, we refer to the resulting technique as *collaborative information processing and routing.*

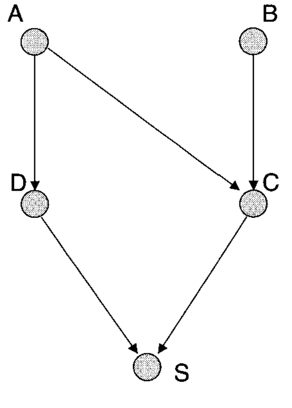


Fig. 2.1 An example network scheme for collaborative information processing and routing.

For example, consider the simple network scheme in Figure 2.1 [100], where vertices A, B, C, D, and S 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 task is to gather and transport information from both A and B to the sink node. While the route from node B to S has one option (BCS), there are two alternative routes from 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 information is strongly correlated, we can take advantage of joint compression to reduce the data volume to be routed. Therefore, the route ACS would be preferred to enable joint compression at node C, given that the gain from reducing data volume to communicate over CS dominates the extra communication cost of choosing ACS instead of ADS. This example indicates the importance of coupling the selection of routing schemes with data compression techniques.

Because sensor readings from the physical world are usually highly correlated, joint compression is particularly useful to reduce data volume by removing redundancy among data from multiple sources. Such joint compression is also referred to as *data aggregation* [100]. 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 data correlation. In many general settings, to choose the optimal routing scheme is NP-hard, and therefore heuristic solutions are preferred. For example, consider the problem with the assumption that a perfect data aggregation can be achieved. In other words, joint compression for an arbitrary number of sources always produces one unit data. Under this assumption, to form a routing tree from a given set of source nodes to a sink node with a minimum number of edges is the Minimal Steiner Tree problem, which is known to be NP-hard[1] [100].

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 depends heavily 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 [117; 135; 156], a unified and widely accepted model remains

[1]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.

elusive. Hence, several research efforts have been attempting to study the relationship between information processing and routing under certain simplifications. For example, the work by Cristescu *et al.* [47] assumes a fixed reduction in volume for any source data that is jointly compressed by data from other sources. Some other models are discussed in Section 2.4.4.

Another important research direction — joint data compression from information coding perspective — deals with the challenge of collaborative information processing and routing. For example, Slepian-Wolf coding [172] 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. With this coding scheme, Cristescu *et al.* show that the routing selection and joint data compression can be perfectly de-coupled [47]. While practical distributed source coding schemes for sensor networks are being developed [96; 181], most existing work for data gathering is based on compression schemes with explicit communication [3; 47; 59; 67; 100; 111; 135; 156; 190]. The work in this book is also based on compression using explicit communication: To perform joint data compression requires the availability of side information from other sources via explicit communication.

Most of the existing work on information processing and routing is intended for a single sink node. When multiple sink nodes are considered, the concept of *network information flow* [4] 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. Integrating these two concepts in the context of WSN applications is an open challenge.

Also, most existing work on collaborative information processing and routing assumes 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 system parameters of real-life scenarios.

2.3 Cross-Layer Optimization for Energy-Efficiency

2.3.1 *Motivation*

A major concern for WSNs is energy-efficiency, which has been addressed by various techniques targeting hardware components [76; 144], media access control (MAC) layer scheduling policies [110; 206], network organization [72; 167], routing protocols [66; 85], signal processing techniques [7; 17], and application level algorithms [168; 170]. Cross-layer optimization is also of particular importance for saving energy in WSNs, since it enables a large space for tradeoffs and the optimization of performance metrics across different layers [147; 162; 212].

Although widely studied in many contexts, there are 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.* In most cases, cross-layer optimization also impacts the system properties or performance at other 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. This 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. Thus, 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 the physical layer, channel access at the MAC layer, routing selection at the routing layer, and sensing coverage at the application layer. Isolating optimization techniques at each individual layer may cause conflicts in optimization goals or even counteracting solutions. It is therefore 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, not widely studied, due to the inherent complexity of coping with multiple performance metrics at multiple layers.

Third, WSNs are usually application specific in terms of the required functionality. The general functionalities supported by strictly layered network/system structure becomes unnecessary, when compared with the layering overhead. Therefore, 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 the 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 useless. We still need a layered structure, so that a clean model is presented at each layer, which abstracts unnecessary details from lower layers. The key point in cross-layer optimization is information sharing through the exposure of certain parameters and the follow-on coordinated optimization across the stack.

A common way to realize cross-layer optimization is to adjust the system performance by tuning low level hardware-based system knobs. One well-known example is the shutdown or sleeping scheduling that tunes the awake time of the sensor nodes to adjust various upper-layer functionalities, including MAC layer scheduling [110; 206], topology control [36; 202], routing selection [201], and coverage for event detection [1; 183]. Besides the motivation to reduce channel contention, and to alleviate the scalability issue by keeping as small a number of sensor nodes awake as possible, the major principle in this case is to deliver "just enough" performance, with minimum resource usage (including energy). Other commonly used system knobs include voltage scaling that adjusts CPU computation time [204], power control that adjusts radio transmission radius [151], and rate adaptation that adjusts radio transmission speed [143]. We will discuss these system knobs in detail in Section 2.4.

2.3.2 *Consideration for Collaborative Information Processing and Routing*

It is quite natural to apply cross-layer optimization techniques to 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
(3) joint information routing and compression across the network

We note that, in many cases, the first two stages also require distributed and coordinated operations among sensor nodes. 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 requirements [1; 183]. Since the sensing and computation capability of each sensor node is limited, signal processing usually requires the coordination of 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 [17]). Thus, for all three operating stages, the cross-layer optimization is expected to be performed in a distributed fashion.

Table 2.1 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	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

From another perspective, cross-layer optimization techniques can be classified by the layer wherein the techniques are applied. For our purpose, we divide the system into 5 layers: hardware layer, physical layer, MAC layer, routing layer, and 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 2.1, we re-interpret the techniques in the context of the above three operating stages of collaborative information processing and routing.

In the next section, we give a brief survey of the system knobs listed in Table 2.1.

2.4 A Brief Survey of Cross-Layer Optimization for Energy-Efficient Collaborative Information Processing and Routing

We present this brief survey based on the system layers where the techniques are applied. Note that although we have narrowed our focus 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 2.1.

2.4.1 *Hardware Layer*

The most important hardware layer technique is low-power circuit design for both the CPU and radio modules, which has been addressed by a large body of literature [124; 144]. 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 [12] 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 [169; 177], 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; shutting down the radio at the MAC layer will be discussed in Section 2.4.3. 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 wake-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 may also be regarded as the energy *vs* sensing coverage tradeoff [1; 183].

Instead of reducing the operating duration of the CPU by the shutdown technique, voltage scaling explores the energy *vs* delay tradeoff by running the CPU at a lowered speed, and therefore a longer operating duration with reduced supply voltage and operating frequency [204]. The key rationale is that the CPU power consumption is quadratically proportional to the supply voltage, with delay being inversely 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 [14; 79; 163; 204] or multi-processor systems [69; 112; 210; 213; 217].

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. The energy cost to transmit one bit is typically around 500 - 1000 times greater than a single 32-bit computation [16; 166]. Thus, for applications with simple functionality, striving for CPU energy-efficiency might not be worthwhile. However, we also envision the development of more advanced, computationally intensive applications within the near future. Moreover, it has been noted that for many high-end sensor nodes, the power consumption of the CPU is around 30-50% of the total power consumption of the system [147], which also motivates energy minimization for CPUs with complex applications.

2.4.2 *Physical Layer*

At the physical layer, energy efficiency is often optimized using techniques such as power control, rate adaptation, and adaptive coding. These techniques 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. These com-

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

plicated 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 transmission delay.

The rationale behind these two tradeoffs can be explained using Shannon's law in wireless communication [46]. 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 pass loss exponent between 2 and 6. Thus, incrementing either the communication radius or data 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 (e.g., the cellular system), to maintain a given level of signal-to-noise quality to compensate for fading effects, thermal noise, and mutual interference in the shared radio spectrum [71; 82]. In the context of WSNs, where multihop communication prevails, power control has been used mainly for determining an appropriate communication radius, which can be either common for all sensor nodes [129], or not [151]. Many research efforts have proposed power control schemes to reduce the communication radius, and thus the power consumption, while achieving global network connectivity [20; 103; 129; 149; 151; 192]. The tradeoff between energy and reliability through power control has also been studied [102].

Rate adaptation (sometimes also referred to as modulation scaling [157]) and adaptive coding were also originally proposed for cellular systems and local wireless networks, with the goal of throughput optimization [77; 188]. The use of these techniques for scheduling packet transmission over a given channel with delay constraint is studied in [143], with an optimal off-line algorithm similar to the one in [204]. The problem is then extended to a star structure with multiple downstream links [64].

Currently, many research efforts are trying to analyze and utilize the joint impact of these two techniques on energy-efficiency and throughput optimization, often with MAC layer scheduling and routing layer decisions [24; 49; 56; 98; 102]. As it has been realized that energy efficiency depends on factors spanning multiple layers, this is becoming a promising research direction.

2.4.3 *MAC Layer*

The MAC layer affects the energy efficiency mainly through the adjustment of transmission scheduling and channel access. A common way to do that is via sleep scheduling [109; 145; 189; 206], from a long time scale perspective, or time-division multiple access (TDMA) [9; 110], from a 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 with Signaling) protocol [145] is a simple improvement over the MACA protocol [94], which turns off the radios of nodes that cannot transmit or receive given the current neighborhood traffic. A more aggressive policy, S-MAC, is proposed by Ye *et al.* [206], 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 [189]. While these works are proposed for a general network topology, a scheduling policy dedicated to a routing tree structure in WSNs is proposed in D-MAC [109]. 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 adaptive sleep scheduling, TDMA provides a stricter, pre-specified sleep scheduling. Tradeoffs between energy and delay, as well as buffering size, are studied by Arisha *et al.* [9]. A novel performance metric of network-wise delay diameter is studied by Lu *et al.* [110].

Another tradeoff of sleep scheduling is the energy *vs* topology, which in turn impacts concurrent transmission scheduling and channel access at the MAC layer, and routing selection at the routing layer. Most of the research efforts along this line try to maintain a backbone style topology of the network, so that the network remains connected with a minimum number of awake sensor nodes [36; 159; 202].

The Span protocol [36] uses a randomized method to elect so-called coordinator nodes to maintain a backbone connectivity with redundancy.

The concept of a virtual grid is proposed in GAF [202], 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 [159], radios are proactivated by using either a paging channel with fixed duty cycle, or a tone on a secondary channel, which provides additional means to explore the energy *vs* latency tradeoff.

2.4.4 *Routing Layer*

The first batch of routing protocols that were adopted for WSNs is mostly based on protocols originally proposed for ad hoc networks, including extensions of AODV and DSR [202]. These routing protocols are still based on traditional, address-centric, peer-to-peer communication patterns, instead of the data-centric paradigm of WSNs.

Directed diffusion [85] is one of the first well-known protocols customized for information routing in data-centric networks. 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 for integrating information processing with routing makes the protocol inappropriate for applications with complex information processing. Some other geographic routing [95] and rumor routing [26] protocols also fall into this category.

The LEACH protocol [73] 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 was first presented by Krishnamachari *et al.* [100]. 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), was investigated by Pattem *et al.* [135]. 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; 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 confirmed by other results. Cristescu *et al.* assumes a simplified compression model [47], 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 TSP, the Shallow Light Tree (STL), is proved to provide 2-approximation performance for minimizing the overall cost of the data gathering tree. Goel *et al.* studied a very similar problem, where the joint entropy was modeled as a concave, but unknown function of the number of source nodes [67]. It is also noted by Goel that the data gathering problem is essentially a single-source buy-at-bulk problem [13]. 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 in [59], which achieves constant factor approximation (in expectation).

Moreover, some research efforts have investigated other routing substrates besides the 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 [78]. 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 [55].

All of these papers [47; 55; 59; 67; 78; 100; 135] assume that certain coding mechanisms are available to accomplish the data aggregation operation, so the authors can focus on a relatively high level algorithm design and performance analysis. 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 [156]. 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 is therefore supportable by the $O(\sqrt{N})$ network capacity. It is also argued by Cristescu *et al.* [47] that when Slepian-Wolf coding can be used without explicit communication, the shortest path tree is 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. We therefore refer to the routing techniques discussed above as *entropy-aware routing*.

Other works have also 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 [107]; the expected number of transmissions (ETX) is used as a path metric for multi-hop transmission [51]; the packet reception rate $\times$ distance metric is used to choose a forwarding node during routing [160]. In this context, various tradeoffs between energy and transmission delay, number of hops, and path length have been explored.

2.4.5 *Application Layer*

At the application layer, both energy-efficient signal processing algorithms [7; 17] and adaptive-fidelity algorithms [171; 182] have been studied for cross-layer optimization. The key idea is to trade the application-level information precision or accuracy for energy.

The tradeoffs between computation and communication were first explored by Acimovic *et al.* [3]. For prediction-based data gathering over a one-dimensional random Gaussian field, such tradeoffs are enabled by adjusting the group size within which prediction is performed — large group increases computation cost but decreases communication cost [3]. Simulation results indicate that the optimal group size increases with its distance from the sink.

Another useful technique at the 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 compression ratio, with larger ratios resulting in longer compression time, and therefore higher energy cost [16; 21]. The use of tunable compression is focused on computationally intensive

Table 2.2 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
	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

applications, where traditional maximal compression becomes less energy-efficient because of the over-paid computation energy for data compression. Thus, carefully choosing the compression ratio is necessary to explore the tradeoffs between computation and communication energy.

2.4.6 *Summary*

In Table 2.2, we illustrate the tradeoffs explored by the these techniques. Note that it is often difficult, if possible, to isolate clearly the different performance metrics involved in the tradeoffs. For example, transmission delay and reliability are closely related at both the physical layer and the 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 the transmission delay and throughput simultaneously. Thus, in many cases, it is necessary and helpful to understand the tradeoffs while taking multiple performance metrics into consideration.

Moreover, the concrete interpretation of a single performance metric may vary across different levels. For example, delay at the application layer often refers to the end-to-end time duration for performing a specific task, e.g., gathering information across the network. At the routing layer, delay usually refers to the time duration of transporting a packet over a path between two sensor nodes. At the physical layer, delay may refer to the time duration for packet transmission over a single link. Furthermore, link-wide delay at the physical layer and path-wide delay at the routing layer also affect system-wide delay at the 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 2.3.1.

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 solutions for other techniques. For example, the radio sleep scheduling at the MAC layer affects network topology, which in turn impacts the routing decision at the routing layer. Also, the sleep scheduling affects channel access at the MAC layer and hence signal interference at the 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 the techniques.

Second, a 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 for the energy *vs* delay tradeoff is how to break the application-level delay constraint into sub-constraints for each individual layer. There is no easy solution for this problem, unless a cross-layer optimization technique can be developed to integrate the tradeoffs at different layers.

This page is intentionally left blank

Chapter 3

Energy Models

3.1 Definitions and Notations

Before describing the energy models, we first give a list of basic definitions and notations that are used throughout the book.

3.1.1 *Mathematics and Graphs*

Let $\mathbb{N}^+$ denote the set of positive natural numbers, i.e., $\mathbb{N}^+ = \{1, 2, \ldots\}$. 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 $\{1, 2, \ldots, i\} \subset \mathbb{N}^+$.

Let $G =< \mathbb{V}, \mathbb{L} >$ denote a graph with vertex set $\mathbb{V}$ and edge set $\mathbb{L}$. 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 book, we often use $u \in \mathbb{V}$ to denote a node, and $e \in \mathbb{L}$ to denote an edge, if the indices of nodes or edges can be ignored in the context.

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

Definition 3.1 Given two nodes V_1 and V_2 in a directed graph $G =< \mathbb{V}, \mathbb{L} >$, 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 3.2 Given two edges L_1 and L_2 in a directed graph $G =< \mathbb{V}, \mathbb{L} >$, 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.

We use the above notations to abstract both the network structure and application tasks of WSNs. When using graphs to represent WSNs, the vertices correspond to the set of sensor nodes, and the edges correspond to the set of communication links between the sensor nodes. Since general network topology is usually modeled as an undirected graph, the predecessor/successor relationship is usually applied in a specific routing substrate, such as a data gathering tree, that is usually represented as a directed graph. For such a data gathering tree, the predecessor/successor relationship defines the order of nodes or edges that a packet may traverse within the tree.

We also use directed graphs to represent the inner-structure of an application. In this case, the vertices correspond to computation tasks, and the edges correspond to communication tasks with directed data flow. Thus, the predecessor/successor relationship specifies the order of execution for the related tasks in the application.

We formally define the above representations in the following two sections.

3.1.2 *Network Topology Graph*

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

Definition 3.3 An *network topology graph* (or simply network graph) $NG =< \mathbb{V}, \mathbb{L} >$ is a graph with vertex set $\mathbb{V}$ representing the set of sensor nodes, and edge set $\mathbb{L}$ representing the set of communication links.

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

Definition 3.4 A *collocated* network of sensor nodes consists of a set of 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 3.5 A *grid* network of sensor nodes consists of a set of nodes placed on a grid structure, where each node can communicate with its eight neighbors (ignoring boundary effects) through one-hop communication.

Definition 3.6 An *arbitrary* network of sensor nodes consists of a set of nodes randomly placed on a planar. Each node can communicate with 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 $\mathbb{V}$, we also associate a weight with each link in $\mathbb{L}$, which reflects the energy cost of transmitting a packet of unit size over the link. In this book, we have different assumptions about the weights according to the specific problem scenarios.

For example, in Chapters 4 and 5, we model the weight as a function of the transmission speed based on the energy model of rate adaptation, which is determined by the physical distance and environment, as well as the radio features. In Chapter 6, we assume that the transmission speed is fixed, and the weight simply becomes a scalar value for each edge, which is determined by the physical distance between the two sensor nodes associated with the link. In other words, the transmission power is adjusted in such a way that just enough reception power is achieved at the receiving node. Recall the signal fading model for wireless communication in Section 2.4.2. This means that the transmission power is proportional to d^{α}, where d is the distance between the sending and receiving nodes and, α is the path loss exponent (between 2 and 6 in most scenarios). Also, in the special case of the grid network in Chapter 6, we assume that the cost of 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 4, 5, and 6, we give a detailed description of the link weights in these chapters accordingly.

3.1.3 *Application Graph*

Given $c \in \mathbb{N}^+$, we use $\mathbb{C}$ to denote a set of c computation tasks, i.e., $\mathbb{C} = \{C_i : i \in [c]\}$. Given $q \in \mathbb{N}^+$, we use $\mathbb{Q}$ to denote a set of q communication tasks between the computation tasks, i.e., $\mathbb{Q} = \{Q_i : i \in [q]\}$. 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*:

Definition 3.7 An *application graph* $AG =< \mathbb{C}, \mathbb{Q} >$ is a directed acyclic graph (DAG) with vertex set $\mathbb{C}$ representing the set of computation tasks, and edge set $\mathbb{Q}$ representing the set of communication tasks.

We often use tasks or activities to refer to both computation and communication tasks. Also, for a communication link, the terms "sending node" and "sender" are used interchangeably, as 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 3.8 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 $Q_k \prec C_j$. Also, we refer to C_i as a *predecessor* of C_j and C_j a *successor* of C_i.

Definition 3.9 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.

This definition of dependency refers mainly to data dependency — the output of C_i acts as an input to 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 node, data exchange is performed through memory exchange, which has a negligible cost compared to packet transmission.

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. 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 $C_i \prec C_j$. Similarly, consider a task C_k with incoming edge Q_i and outgoing edge Q_j: We have $Q_i \prec C_k$ and $C_k \prec Q_j$, implying $Q_i \prec Q_j$.

Note that we do not model control dependency with the above application graph. This is reasonable if we consider the worst case execution scenario and abstract computation tasks at a reasonably coarse granularity. Similar models are widely used in parallel and distributed computing [27; 68; 203] and real-time computing [63; 79; 93; 112].

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. Based on the specific problem scenarios, we give more detailed descriptions and notations of task weights later.

The application graph can be used to represent both signal processing algorithms, and information processing and routing procedure. For example, the recursive, one-dimensional Fast Fourier Transformation (FFT) algorithm is given in Figure 3.1(a), with an example task graph of 4 points demonstrated in Figure 3.1(b).

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 task graph shown in* Figure 3.2, each task is a data aggregation operation. Note that a general application graph for information processing does not need to be a tree structure.

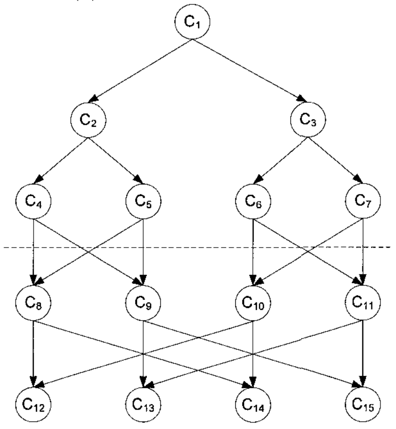
The task graph for the operation of information processing and routing, without a given routing tree, is trickier. Basically, the structure of the task graph cannot be determined until the structure of the routing tree is determined. In this case, the task graph is insufficient to describe the application.

3.1.4 *Performance Metrics*

We are interested in several performance metrics. Energy is our primary focus. We usually use ϵ_i to denote the energy cost of either a computation or communication task, where i is the corresponding index of the task in the application graphs. When we consider the scenario of 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. The subscription i is often ignored if it can be inferred from the context.

```
FFT($A$, $\omega$)
1. Set $l$ = 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.     $Y[i] = Y^{(0)}[i] + \omega^i \times Y^{(1)}[i]$
7.     $Y[i + l/2] = Y^{(0)}[i] - \omega^i \times Y^{(1)}[i]$
8. Return Y
```

(a) sequential algorithm



(b) example application graph with 4 points

Fig. 3.1 Fast Fourier Transformation (FFT) algorithm.

In real-time scenarios, the accumulated time cost of tasks is often limited by a certain constraint at the application level. We refer to such a constraint as the *latency constraint*, denoted as Γ. Such a latency constraint is common in the literature of real-time task scheduling [63; 79; 93; 112].

The notion of latency constraint is usually used together with the notion of periodicity in real-time task scheduling. An application is periodic if it needs to be repeated 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 and transient applications. For transient applications, the latency

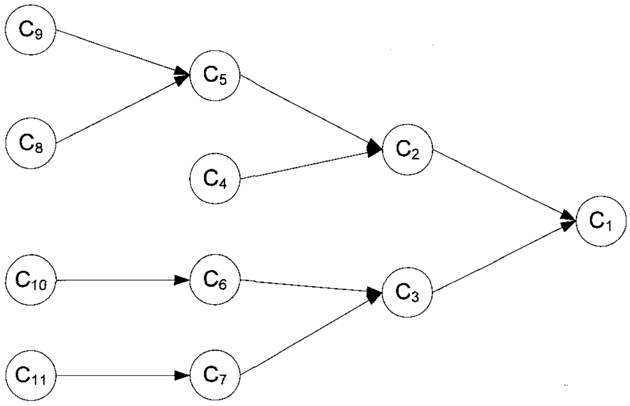


Fig. 3.2 Application graph for information transportation over a given tree.

constraint is easy to understand. For periodic applications, we consider the latency constraint to be less than the application period. Otherwise, certain pipeline techniques need to be developed so that the execution of two or more rounds of information processing or routing overlap. This is beyond the scope of our research.

A similar concept involving periodic applications in WSNs is the epoch-based system [115]. Within each epoch, the desired system functionality (e.g., a query) needs 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., [57; 65; 179]).

Some other performance measurements, such as throughput, have also been studied [78]. Our research is focused on applications that require low-duty cycle, for which throughput optimization is not a concern.

3.2 Energy Models

In this section, we describe the energy models for the three system knobs under consideration. For voltage scaling, the model is developed at the circuit switching level. For rate adaptation, the model is developed at the physical level. For tunable compression, the model is developed at the application level. However, from the perspective of optimization at a relatively high abstract level, we are more interested in the tradeoffs involved in these knobs, rather than the details of the energy models and the corresponding low level implementation. 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. We discuss these energy functions in detail in the following sections.

3.2.1 *Voltage Scaling*

Consider a computation task. There are three major components of power consumption for executing the task by a CMOS integrated circuit: switching power, short-circuit power, and leakage power [35]. We focus on the switching power, which dominates the power consumption in most cases.

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:

$$P_{sw} = P_t \times C_L \times V_{dd}^2 \times f_{clock} \,. \tag{3.1}$$

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

From (3.1), we observe that the power consumption increases quadratically with supply voltage. Therefore, the power consumption can be reduced by decreasing the supply voltage. However, this comes at the expense of reduced processing speed, and therefore increased processing time τ, which is given by:

$$\tau = k_c \frac{V_{dd}}{(V_{dd} - V_T)^2} \,, \tag{3.2}$$

where k_c is a constant and V_T is the threshold voltage [35].

Based on the above models, we can see that $P_{sw} \propto V_{dd}^2$ and $\tau \propto \frac{1}{V_{dd}}$. Thus, 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 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 [30], which is detailed in Chapter 4.

Note that switching the voltage is not free: It costs both time and energy. While most of the techniques proposed for voltage scaling ignore this fact, or assume that such time and energy costs are absorbed in the execution cost of tasks, there are a few papers that explicitly address these costs as well [79]. Also, while the switching power is being continuously reduced by advances in CMOS technology, the leakage power becomes a significant portion of the total power consumption. Various techniques have been proposed to reduce leakage current, such as scaling the threshold voltage [118; 130], and power supply gating [128]. The tradeoffs between voltage scaling and shutdown policy has also been studied [87]. Although we focus solely on switching power in this book, our techniques can be extended to incorporate leakage power based on the work by Jejurikar *et al.* [87].

3.2.2 *Rate Adaptation*

We model the transmission energy using the example of modulation scaling [157], based on the Quadrature Ampitude Modulation (QAM) scheme [188]. 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 [157]:

$$\tau = \frac{s}{b \cdot R_s} , \tag{3.3}$$

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 [157], denoted as $w(\tau)$:

$$w(\tau) = [C_{tr} \cdot (2^{\frac{s}{\tau \cdot R_s}} - 1) + C_{ele}] \cdot \tau \cdot R_s , \tag{3.4}$$

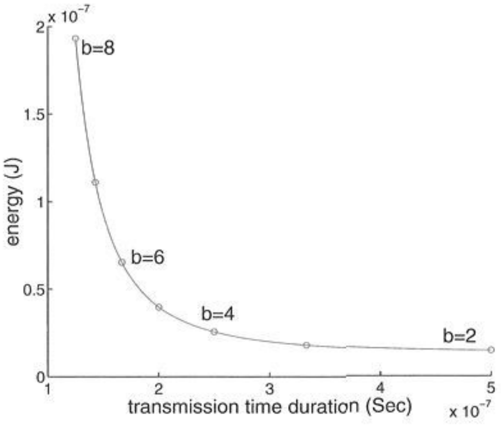
where C_{tr} is determined by the quality of transmission (in terms of Bit Error Rate) and the noise power, and C_{ele} is a device-dependent parameter that determines the power consumption of the electronic circuitry of the sender. Further, the output power, P_o, and the electronic power, P_e, can be modeled as follows:

$$P_o = C_{tr} \cdot R_s \cdot (2^b - 1) \text{ and} \tag{3.5}$$
$$P_e = C_{ele} \cdot R_s\ . \tag{3.6}$$

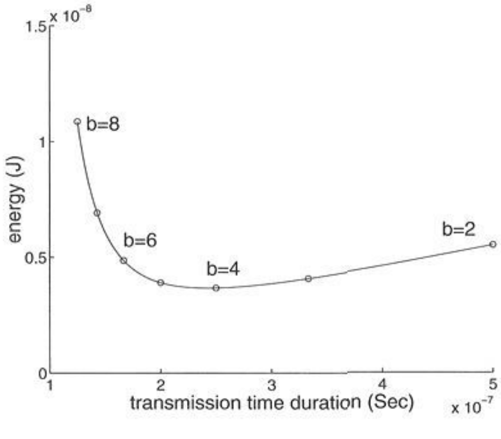
Note that different assumptions about the radio characteristics, including power consumption and data rate, may significantly affect the analysis of various energy-saving mechanisms. In this work, we consider the radio modules described in [147; 191]. Typically, for *short-range* communication with $R_s = 1$ Mbaud, the electronic power of the radio is approximately 10 mW, while the output power is approximately 1 mW at 4-QAM (which translates into 2 Mbps). Note that the data rate and power consumption are better than currently available radios for commercial sensor nodes, which typically support a data rate up to 100 Kbps with slightly higher power characteristics, such as the Berkeley motes. However, radio devices with these 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 [72], 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 radii. Specifically, we set the communication range to 30 m, implying an output power of $10 \text{ pJ/bit/m}^2 \times 30^2 \text{ m}^2 \times 2 \times 10^6 \text{ bit/sec} = 18$ 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 purposes only, i.e., 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.

Figure 3.3 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 observe a 10-fold energy reduction for long-range communication by varying b from 8 to 2, and a 3-fold energy reduction for short-range communication by varying b from 8 to 4. Therefore, it is more beneficial to explore the energy-latency tradeoffs for the long-range communication.



(a) Long-range communication



(b) Short-range communication

Fig. 3.3 Energy-latency tradeoffs for transmitting one bit.

Although QAM is used as an example to abstract the energy model, the presented algorithms extend to other modulation schemes and techniques that can be used to trade latency against energy, such as code scaling [10].

3.2.3 *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 compression ratio, with larger compression ratios resulting in longer compression times, and therefore higher energy cost [16; 21].

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 of data is inversely proportional to the output size. Further, the energy cost is proportional to the time complexity [16]. We illustrate this rationale using the example of *gzip* to compress the benchmark file "alice29.txt" from the Canterbury Corpus [21], 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 3.4(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 [16].

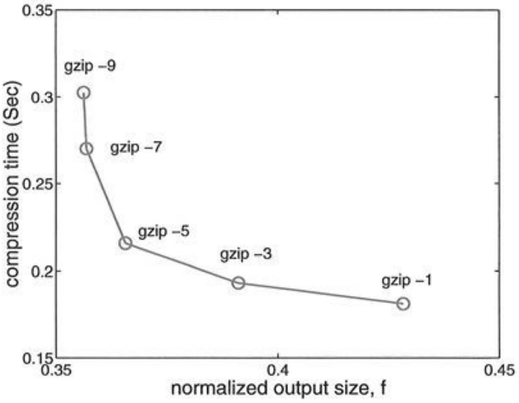
We define a pre-specified system parameter, $\gamma \geq 0$, which abstracts the relative energy cost of compressing one unit of data normalized by the cost of communicating one unit of data. Following the above rationale, and for the purpose of illustration, the energy cost of compressing a data packet of size s to an output of size f is modeled as function

$$g(f) = s\gamma(\frac{s}{f}) \, . \tag{3.7}$$

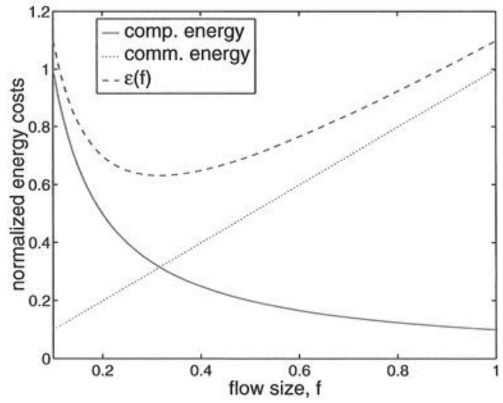
While the first term s indicates that the energy cost is proportional to input size, the term $\frac{s}{f}$ signifies that the energy cost is also inversely proportional to the compression ratio.

We now illustrate the tradeoffs between computation and communication energy using the example of a one-hop link. Let $e = (V_1, V_2)$ denote the link, where V_1 generates one unit of data that needs to be transmitted to V_2, after appropriate compression by V_1. 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:

$$\epsilon(f) = \frac{\gamma}{f} + f \cdot w_e \, . \tag{3.8}$$



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



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

Fig. 3.4 Energy tradeoffs by tunable compression.

We plot $\epsilon(f)$ in Figure 3.4(b), with $w_e = 1$, $\gamma = 0.1$ and varying f from 0.1 to 1 (we omit the boundary effect of ρ for now). Intuitively, $w_e = 1$ means that transmitting one unit of data costs one unit of energy. Since the energy of transmitting one bit is typically comparable to the energy of executing 500 - 1000 instructions with contemporary hardware [16; 166], the realistic meaning behind $\gamma = 0.1$ is that around 50 - 100 instructions need to be executed to generate 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 4

Information Processing within a Collocated Cluster

4.1 Overview

4.1.1 *Motivation*

While information processing in wireless sensor networks is distributed throughout the network, a practical strategy is to organize geographically proximal nodes into small groups as basic units for the 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 [72; 208], each 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 [42] and the Fast Fourier Transformation (FFT) [44].

In this chapter, we consider clusters consisting of a collocated network of sensor nodes. We consider a general scenario where 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 three components to specify the resulting execution of the application: (1) the *assignment* of computation tasks to sensor nodes and communication tasks to channels, (2) the *voltage settings* for computation tasks and the *data rate setting* for communication tasks, and (3) the *scheduling* of computation and communication tasks. We refer to the combination of instantiations of these three components as a *task allocation*.

Besides latency, 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. We also consider the *task placement constraint*, which is typically required when certain tasks for sensing the raw data must be allocated to different sensor nodes.

We assume that sensor nodes are completely shut down in the idle state when there are no computation and communication tasks to be executed. Thus, we only need to consider the energy cost for executing computation and communication tasks. To synchronize the communication between sensor nodes, we assume that sleeping sensor nodes can be awakened for packet transmission by using a full duty cycle, ultra-low power wakeup radio with almost no delay or energy penalties. One promising technique to realize wakeup radios is discussed by Zhong *et al.* [215].

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

4.1.2 *Technical Overview*

We propose the concept of energy-balanced task allocation for information processing with a cluster of collocated sensor nodes. As we shall see in Section 4.2, most of the previous efforts in energy-aware task allocation or resource management try to minimize the overall energy dissipation 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 some time. 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 computation and communication. We 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 [106], though the running time can be long. While this ILP formulation can be used as a performance benchmark for small scale problems, we also propose a polynomial time 3-phase heuristic for solving large scale problems.

Our simulation results show that for small scale problems, up to 5x lifetime improvement is achieved by the ILP-based approach, compared to 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 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.

4.1.3 *Chapter Organization*

We discuss related work in Section 4.2. The energy-balanced task allocation problem is defined in Section 4.3. The ILP formulation of the problem is given in Section 4.4. The 3-phase heuristic is described in Section 4.5. Simulation results are demonstrated in Section 4.6. In Section 4.7, we give a summary of the chapter.

4.2 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 [14; 79; 163; 204]. Recently, research interests have shifted to multiprocessor systems. A list-scheduling based heuristic is proposed by Gruian and Kuchcinski to recalculate dynamically the priority of communicating tasks [69]. Also, static and dynamic variable voltage scheduling heuristics for real-time heterogeneous embedded systems are proposed by Luo and Jha [112]. An approach based on critical-path is used for selecting the voltage settings of tasks. However, both papers assume that the task assignment is given. A similar problem to the one studied in this paper is investigated by Zhang *et al.* [213]. A two-phase framework is presented to determine first the allocation of tasks to processors and then the voltage settings of tasks using convex programming. In the work by Zhu *et al.* [217], a dynamic processor voltage adjustment mechanism for a homogeneous multi-processor environment is discussed. However, the time and energy costs for communication activities are not addressed in most existing work.

The goal of all the above research efforts is to minimize the overall energy dissipation of the system. While this goal is reasonable for tightly coupled systems, it does not capture the nature of WSNs. The reason is that to minimize the overall energy dissipation can lead to heavy use of energy-effective sensor nodes, regardless of their remaining energy. The consequently short lifetime of such sensor nodes will very likely hinder the system in delivering required performance. This weakness is a major motivation for the proposed energy-balanced task allocation.

Our work considers the energy and time costs of both computation and communication activities. As indicated by several research efforts, 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-balancing in distributed computing. However, the communication activities over the same wireless channel need to be serialized so that run-time contentions can be avoided. The serialization imposes new challenges that distinguish our problem from most of the existing works for load-balancing or real-time scheduling in distributed systems.

4.3 Problem Definition

4.3.1 *System Model*

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

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 3.2.1, 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 their size and instruction components, such a function may vary for different tasks

We also consider discrete transmission rates for wireless communication. This is practical, since the modulation setting in (3.4) 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 3.2.2, 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 3.2.2. 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 models can be found in Chapter 3, it suffices to understand the problem and approach in this chapter simply based on these 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) activities over the same sensor node (wireless channel) cannot overlap with each other.

For ease of analysis, we assume that the processors and radios are completely shut down in the 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 processor/radio can be larger than the energy cost for keeping the processor/radio on during the idle state. Methods for determining the optimal decision during the idle state for energy savings can be found in [170], and is beyond the scope of this book. We assume that sleeping sensor nodes can be awakened for packet transmission using a full duty cycle, ultra-low power wakeup radio, which can be realized using the techniques proposed by Zhong *et al.* [215]. In the case of heavy-duty computation and communication applications, as considered in this chapter, it is fair to ignore the delay and energy penalties for wakeup radios.

4.3.2 *Application Model*

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

We consider applications that collect single-modal information. For such applications, it does not make sense to place more than one source task on the same sensor node. Thus, we define a *task placement constraint*: no two source tasks can be assigned to the same sensor node. Nevertheless, our model and approach are extendible to general cases where any pair of tasks must be or must not be assigned to the same sensor node.

For any task $C_i \in \mathbb{C}$, let W_i denote its workload in terms of the worst-case number of required computation cycles. Based on our discussion in Section 4.3.1, we can calculate the processing delay and energy cost for C_i at each voltage level. Let τ_{ij} denote the processing delay 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 any task $Q_i \in \mathbb{Q}$, 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 values of τ_{ij}, ϵ_{ij}, τ'_{ij}, ϵ^s_{ij} and ϵ^r_{ij} can be calculated based on the system and application models. Thus, we assume they are given *a priori* in the following discussion.

4.3.3 *Task Allocation*

Based on the above system and application models, a *task allocation* is defined as (1) the *assignment* of computation tasks to sensor nodes and communication tasks to channels, (2) the *voltage settings* of computation tasks and the *data rate setting* of communication tasks, and (3) 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 activity 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 duration from when the application starts, 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 during each period. Let $\mathcal{E}_i$ denote the energy dissipation of $V_i \in \mathbb{V}$ 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 the 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 chapter. 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:

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

4.4 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 4.1.

To capture the relative order imposed by the precedence constraints among activities, we define the Constraint Set 1 shown in Figure 4.1. It

Table 4.1 List 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
$\tau'_{ij}, \epsilon^s_{ij}, \epsilon^r_{ij}$	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 between 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
$\{q_{ij}\}$	A set of 0-1 variables such that q_{ij} equals one iff the transmission of Q_i finishes before Q_j starts

is easy to verify that the exclusive access constraint for activities with precedence constraints is also enforced by Constraint Set 1. However, for activities that do not have precedence constraints between them, an extra set of constraints are needed (Constraint Set 2 in Figure 4.2), to enforce the exclusive access constraint. In addition, the task placement constraint is captured by the Constraint Set 3 in Figure 4.2.

The complete ILP formulation is given in Figure 4.3, where $\mathcal{E}$ is an auxiliary variable. In the figure, the term $\sum_{C_i \in \mathbb{C}}\{x_{ik} \sum_j (y_{ij}\epsilon_{ij})\}$ gives the energy cost for computation tasks on V_k. The term $\sum_{Q_i=(a,b)\in\mathbb{Q}}\{x_{ak}(1 - x_{bk}) \sum_j (u_{ij}\epsilon^s_{ij}) + (1 - x_{ak})x_{bk} \sum_j (u_{ij}\epsilon^r_{ij})\}$ 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 using standard linearization techniques [196]. Due to space limitations, we omit the details of linearization.

Constraint Set 1:

$\forall C_i \in \mathbb{C}$	
$\sum_j x_{ij} = 1$	// 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
$\alpha(i) \geq \max_{Q_l=(j,i)\in\mathbb{Q}}\{\delta(l)\}$	// 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
$\forall C_i, C_j \in \mathbb{C}$	
$r_{ij} = 1$ 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 node
$\forall Q_i = (a, b) \in \mathbb{Q}$	
$\sum_j z_{ij} = 1$	// 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

Fig. 4.1 Constraint Set 1 for the ILP formulation.

4.5 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, the tasks are grouped into clusters with the goal of minimizing the overall execution time of the application. In the second phase, task clusters are assigned to sensor nodes in such a way that the highest energy dissipation among all sensor nodes, normalized by their remaining energy, is minimized. In the last phase, the

Constraint Set 2:

$\forall C_i, C_j \in \mathbb{C}$, 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 $p_{ij} = 1$

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

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

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

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

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

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

Constraint Set 3:

$\forall C_i, C_j \in \mathbb{C}$, 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

Fig. 4.2 Constraint Sets 2 and 3 for the ILP formulation.

system lifetime is maximized by lowering the voltage levels of tasks. The details of the heuristic are as follows.

4.5.1 *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 activities in order to reduce the overall execution time of the application.

Minimize $\mathcal{E}$
Subject to

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

and Constraint Sets 1, 2, and 3

Fig. 4.3 ILP formulation for the energy-balanced task allocation problem.

The idea of Phase 1 is similar to the algorithm proposed in [152]. However, traditional approaches for task clustering usually assume a full, wired connection among processors, so that all communication can be parallelized, whereas in our problem, communication activities 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 activities ready at different times. Activities 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.

```
Begin
1. Each task is assumed to constitute a cluster by itself
2. Set $\mathbb{Q}$ as the list of communication tasks in a non-decreasing order
   of their weights
3. $\Phi \leftarrow$ Travese()
4. While $\mathbb{Q}$ is not empty Do
5.      Remove the first edge from $\mathbb{Q}$, 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
End
```

Fig. 4.4 Pseudo code for Phase 1.

The pseudo code for Phase 1 is shown in Figure 4.4. 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, with respect to 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 therefore Φ.

The pseudo code for Traverse() is shown in Figure 4.5. In the code, we maintain a queue of activities, A, which stores all the ready computation or communication activities 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 activities in Lines 7, 13, and 14. In Lines 9 and 16, the timestamps are updated based on the execution time of the scheduled activities. The actions in Lines 17 and 18 ensure that the radio can be tuned to at most one channel at any time.

4.5.2 *Phase 2*

In this phase, we assign the task clusters from Phase 1 to the actual sensor nodes in $\mathbb{V}$. Note that multiple clusters can be assigned to the same sensor node. Based on the contained tasks and the corresponding communication activities, we first calculate the energy cost of each cluster. Let m denote the number of 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 dissipation of π_i. The normalized energy dissipation (*norm-energy* for short) of a sensor node is given as the sum of the energy dissipation 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 4.6. Initially, π is sorted into a non-increasing order of clusters, by energy dissipation. 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, the function TraverseAssigned() is used to find the execution time of the application, based on the resulting assignment.

```
Begin
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 activity from $A$
6.     If the removed activity is a computation task, denoted as $C_i$
7.         Set $\alpha(i) \leftarrow \max\{$ready time of $C_i$, timestamp of $CL(i)\}$
8.         $\beta(i) \leftarrow \alpha(i) + \tau_{i1}$
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 $Q_i = (C_a, C_b)$ denote the removed communication task
13.        Find the channel with the smallest timestamp, say the $j$-th
           channel
14.        Set $\gamma(i) \leftarrow \max\{$ready time of $Q_i$, timestamp of the $j$-th
           channel$\}$
15.        $\delta(i) \leftarrow \gamma(i) + \tau'_{i1}$
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 activities 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
End
```

Fig. 4.5 Pseudo code for function Traverse().

TraverseAssigned() differs from Traverse() in that, in Line 7 of Figure 4.5, each computation activity is scheduled on the sensor node to which it is assigned. Thus, timestamps are maintained for all sensor nodes, instead of clusters.

4.5.3 *Phase 3*

In this phase, the voltage levels of computation tasks and the data rates of communication tasks are adjusted to maximize the system lifetime. An

```
Begin
1. Sort $\pi$ in a non-increasing order of the energy dissipation of clusters
2. While $\pi$ is not empty Do
3.      Choose the first element, $\pi_1$, in $\pi$
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 $\pi_1$ to the sensor node that gives the minimal expected
        norm-energy
6.      Update the norm-energy of the sensor node
7.      Remove $\pi_1$ from $\pi$
8. $\Phi \leftarrow$ TraverseAssigned()
9. If $\Phi > \Gamma$, Return failure
End
```

Fig. 4.6 Pseudo code for Phase 2.

iterative greedy heuristic is used (shown in Figure 4.7). Let $\mathcal{E}$ denote the maximum 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 that, by decrementing its current voltage level, $\mathcal{E}$ can be decreased the most. The increased latency caused by decreasing the voltage or rate level is added to Φ. Since the schedule of activities can be changed by the latency increment, Φ is re-computed by traversing the DAG every time it reaches Γ (in Line 15).

For each $V_i \in \mathbb{V}$, 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 with t_j, which indicate the reduction in energy cost and increment in latency if the voltage or transmission rate of t_j is decremented. 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.

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 4.7, 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 to the sending energy.

Time Complexity Analysis: In Phase 1 (Figure 4.4), the **While** iter-

```
Begin
1. For each $V_i \in \mathbb{V}$, 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
End
```

Fig. 4.7 Pseudo code for Phase 3.

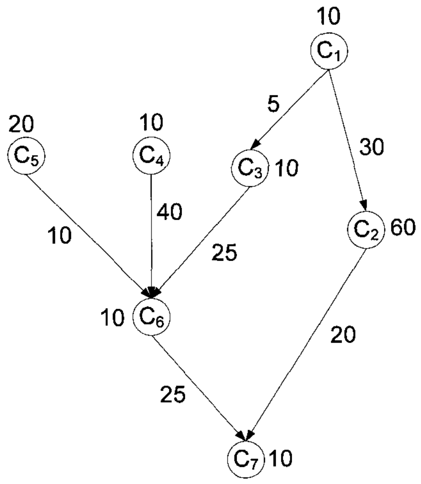
ation is executed q times (recall that q is the number of communication tasks). Function Traverse() in Line 6 takes $O(c + q)$ time (recall that c is the number of computation tasks). Thus, Phase 1 needs $O(q(c + q))$ time. In Phase 2 (Figure 4.6), the ordering in Line 1 takes $O(m \log m)$ time (recall 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 traversal in Line 8 takes $O(c+q)$ time. Thus, Phase 2 takes $O(m \log m + vm + c + q)$ time. In Phase 3 (Figure 4.7), the sorting in Line 1 takes $O((c+q) \log(c+q))$ time. The number of voltage switches in Line 11 is bounded by dc (recall that d is the number of voltage options). The number of rate switches in Line 14 is bounded by fq (recall that f is the number of available rate options). To update ED_r in Line 10 requires $O(\log(c+q))$ time. Let ϕ denote the number of times TraverseAssigned() is called in Line 15. The time complexity of Phase 3 is $O((c+q)\log(c+q)+(dc+fq)\log(c+q)+\phi(c+q))$ $= O(dc+fq)\log(c+q)+p(c+q)$. 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 $p \leq dc+fq$, the worst case time complexity can be simplified to $O((dc+fq)(c+q+\log(c+q))+(q+v)c)$. Assuming that the number of voltage and rate options are fixed for given sensor node hardware, the running time scales quadratically with the number of tasks and linearly with the number of sensor nodes.

An Example: We illustrate the execution of the above heuristic with a simple example. We assume a cluster of 3 sensor nodes connected by 2 channels. Each sensor node has 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 units to transmit one data unit over any channel. The application graph is shown in Figure 4.8(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 4.8(b). We assume that $\Gamma = 250$ time units.

In Phase 1, the voltage levels of all tasks are set to D_h. The sorted edge list with respect to edge weights is $\{(C_4,C_6),(C_1,C_2),(C_3,C_6),(C_6,C_7),$ $(C_2,C_7),(C_5,C_6),(C_1,C_3)\}$. Table 4.2 traces the execution of Phase 1, where Φ_i is the execution time of the application at the completion of step i. The clustering steps are also illustrated in Figure 4.9. The sub-figures (a) through (d) correspond to the application graph at the completion of steps 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.

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 $\pi_3 = \{C_5\}$. Since the remaining energy for



(a) Application graph

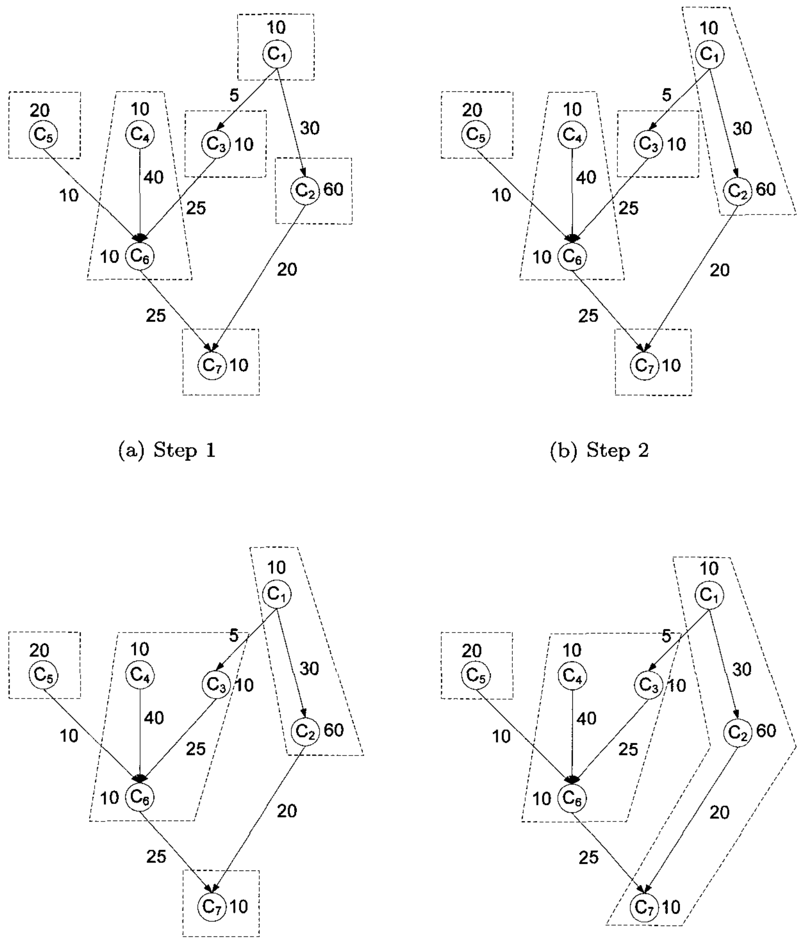
Task	Time cost		Energy cost	
	D_h	D_l	D_h	D_l
C_1	10	33	20	6
C_2	60	199	120	36
C_3	10	33	20	6
C_4	10	33	20	6
C_5	20	66	40	12
C_6	10	33	20	6
C_7	10	33	20	6

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

Fig. 4.8 An application example.

each of the three sensor nodes is 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 the 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,



(a) Step 1 (b) Step 2

(c) Step 3 (d) Step 4

Fig. 4.9 Clustering steps for the application in Figure 4.8.

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 decrease the norm-energy of the critical sensor node from 0.19 to 0.1, implying a system lifetime improvement by approximately a factor of 2.

Table 4.2 Trace of clustering steps in Figure 4.9.

Step i	Edge examined	Φ 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

4.6 Simulation Results

A simulator, based on the system and application models presented in Section 4.3, was developed to evaluate the performance of our approach, using application graphs from a synthetic approach, as well as 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 activities compared to computation activities, 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 standalone 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 10% (or better) precision is achieved.

4.6.1 *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.

4.6.1.1 *Simulation Setup*

The structure of the application graph was generated using a method similar to the one described in [53]. The only difference was that we enforced multiple source tasks in the generation of the DAG.

Since we were concerned with computation-intensive applications, we extracted parameters about the energy cost of computation and communication from the Rockwell Science Center (RSC) WINS node [197]. The power consumption of an Intel StrongARM 1100 processor with 150 MIPS was approximately 200 mW. This implied that the time and energy costs per instruction were about 5 nSec and 1 nJ, respectively. Also, the power of the radio module used in WINS was 100 mW at 100 Kbps, implying that the time and energy costs for transmitting a bit were approximately 10 μSec and 1 μJ, respectively. We set the parameters for our simulator so that the time and energy costs for computation and communication activities roughly followed the above data.

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 was assumed that other levels of computation speed were 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 [30; 121], the power consumption of the processor was modeled as a polynomial function of processing speed, $g(SP)$ of at least degree 2. Thus, 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 [121], 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 at the highest speed, and 6.7 mSec and 1 mJ at the lowest speed.

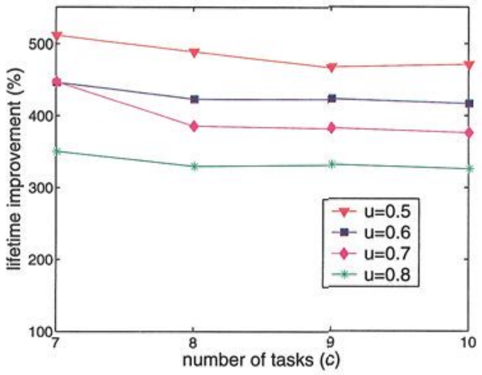
The time and energy costs of communication activities were determined by the number of data units to transmit and the values of τ and ε. Based on the data for WINS, we assumed that the time for transmitting one bit was 10 μSec, and the corresponding energy cost was 1 μJ. To focus on the main issues, we set the startup energy dissipation of the radio to be zero. To study the effect of different communication loads with respect to the computation load, the number of bits per communication activity followed a uniform distribution between $200CCR(1 \pm 0.2)$, where CCR (communication to computation ratio) was a parameter indicating the ratio of the average execution time of the communication activities to that of the computation activities. Intuitively, a larger value of CCR implied a relatively heavier communication load compared to the computation loads. Note that by varying CCR, we abstracted 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 $[0, 20]$.

The period of the application, Γ, was generated in the following way. We first defined 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 were then divided into layers, with nodes in each layer having the same distance. Since the average time to execute a task at the highest speed was 2 mSec, the computation time required for a layer was estimated to be $2\lceil \frac{\kappa}{v} \rceil$ mSec, where κ was the number of tasks in the layer. By doing so, we implicitly assumed full parallelism in executing the tasks at each layer. In addition, the expected number of communication activities initiated by a task was estimated as its out-degree minus 1. Assuming there were in total η communication activities requested by all the tasks in a specific layer, the corresponding time cost was estimated to be $2CCR\lceil \frac{\eta}{K} \rceil$ mSec. Γ was then set to the sum of the computation and communication time cost of all layers over u, where $u \in [0, 1]$ was a parameter that approximates the overall utilization of the system. The setting of u was important, as it determined the latency laxity for trading against energy. Intuitively, a larger value of u implied a tighter latency constraint, and thus less latency laxity.

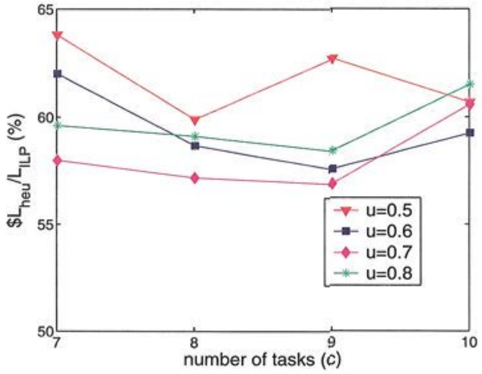
The remaining energy of the sensor nodes followed a uniform distribution between $E_{mean}(1 \pm 0.3)$, where E_{mean} was a fairly large number.

4.6.1.2 *Small Scale Problems*

We first conducted simulations for small scale problems, with 3 sensor nodes, 3 voltage levels, 2 channel, and 7 - 10 computation tasks. The number of source tasks in the application graph was set to 2, while the maximal in-degree and out-degree for each node were set to 3. A commercial software package, LINDO [106], 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 had not yet been found. Then, the best solution obtained in that time was returned. We observed that in most cases, LINDO was able to find the optimal solution within two hours.



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



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

Fig. 4.10 Lifetime improvement of our approaches for small scale problems (3 sensor nodes, 3 voltage levels, 2 channels, CCR = 1).

The data shown in Figure 4.10 is averaged over more than 70 instances. In Figure 4.10(a), we illustrate the lifetime improvement achieved by the ILP-based approach, which was calculated as $\frac{LT_{ILP}}{LT_{raw}} - 1$. We can observe an improvement around 3x - 5x. Figure 4.10(b) shows the performance ratio of the 3-phase heuristic over the ILP-based approach, i.e., $\frac{LT_{heu}}{LT_{ILP}}$. We 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 was approximately zero, the average running time of the ILP-based approach ranged from 550 Sec ($c = 7$, $u = 0.5$) to 5900 Sec ($c = 10$, $u = 0.8$) on a Sun Blade1000 machine with an UltraSparc III 750 Mhz CPU.

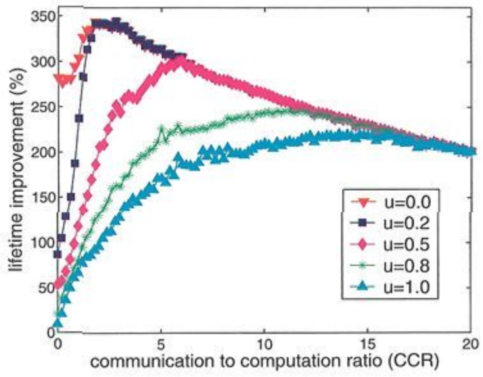
4.6.1.3 *Large Scale Problems*

A set of simulations was conducted to evaluate the performance of the 3-phase heuristic for problems with 10 sensor nodes, 8 voltage levels, 4 channels, 60 - 100 computation tasks, $CCR \in [0, 20]$, and $u \in [0, 1]$. The number of source tasks in the application graph was set to 6. The maximal in-degree and out-degree for each node were set to 5. Due to the large size of the problems, it was impractical to obtain the optimal solutions by using the ILP-based approach. Thus, we used the lifetime improvement achieved by the 3-phase heuristic as the evaluation metric, which was calculated as $\frac{LT_{heu}}{LT_{raw}} - 1$. The simulation results are shown in Figure 4.11.

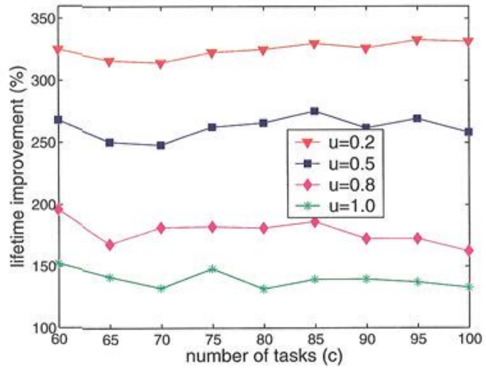
An improvement of up to 3.5x in the system lifetime is observed in Figure 4.11(a). We can see that the improvement increased when u decreased, as the latency laxity increased accordingly. The lifetime improvement saturated when u approached 0, i.e., the latency constraint approached ∞. The curve with $u = 0.0$ gives the upper bound of the improvement that could be achieved by our heuristic with respect to variations in CCR.

The effect of CCR was more complicated. For example, when $u = 0.5$, the lifetime improvement increased when $CCR \leq 6$ and decreased when CCR went beyond 6. This was because, when CCR was small, the computation activities dominated the overall energy costs of the application. By increasing CCR, we actually increased the latency constraint without increasing the computation load, which in turn could be traded for lifetime improvement. However, when CCR reached some threshold value, the communication energy cost became more significant than that of the computation activities. Thus, the lifetime improvement achieved by reducing computation energy became limited. We shall see later that this shortcoming can be overcome by incorporating modulation scaling into our heuristic.

Figure 4.11(b) shows the lifetime improvement with the number of computation tasks, c, varying from 60 to 100. We see that the performance of our approach was quite stable with respect to variations in c.



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



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

Fig. 4.11 Lifetime improvement of the 3-phase heuristic for large scale problems (10 sensor nodes, 8 voltage levels, 4 channels, 60-100 tasks).

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 were guaranteed to have feasible solutions. We observed that the miss rate of the 3-phase heuristic was significant only when CCR was close to zero. Thus, we show the miss rate with $CCR = 0$ in Figure 4.12. Also, the running time of the heuristic was approximately 0.5 mSec on a Sun Blade1000 machine with an UltraSparc III 750 Mhz CPU.

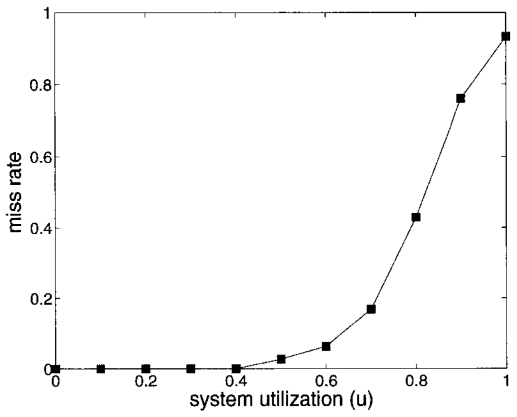


Fig. 4.12 Miss rate of the 3-phase heuristic (10 sensor nodes, 8 voltage levels, 4 channels, 60 computation tasks, $CCR = 0$).

4.6.1.4 *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 computation tasks, 4 channels, $CCR = 2$, $u \in \{0.2, 0.5, 0.8, 1.0\}$, and 1 to 10 voltage levels. The results are presented in Figure 4.13.

The plots show that when $u > 0.2$, the performance of the heuristic could be significantly improved by increasing the number of voltage levels from 1 to 4. Further increase in the number of voltage levels did 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 largest 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 was so large that the voltage level of most computation tasks could be set to the lowest option. Thus, there was almost no improvement by increasing the number of voltage levels beyond 2.

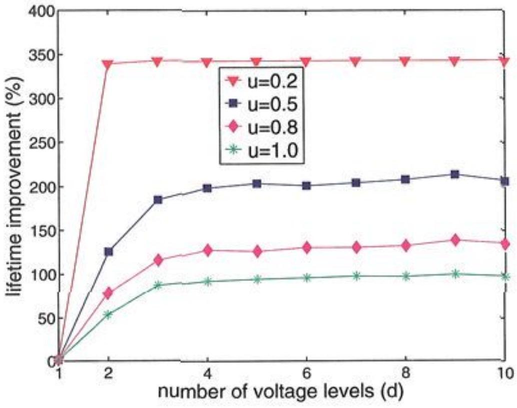


Fig. 4.13 Impact of variation in number of voltage levels (10 sensor nodes, 4 channels, 60 computation tasks, $CCR = 2$).

4.6.1.5 *Incorporating Rate Adaptation*

We used modulation scaling in Section 3.2.2 to illustrate the incorporation of rate adaptation. Due to the underlying single-hop connection, we assumed that all sensor nodes had the identical settings for parameters C_{tr}, C_{ele}, and R_S. From [157], we set $C_{ele} = 10^{-7}$. To investigate the impact of different energy/time ratios for data transmission, we set $C_{tr} = 10^{-7}$ and 10^{-6} as two different simulation scenarios. The modulation level, b, was set to even numbers between 2 and 6. For a fair comparison with the results in Section 4.6.1.3, we set $R_s = 1.7 \times 10^4$, so that when $b = 6$, it took roughly 10 μSec and 1 μJ to transmit a bit when $C_{tr} = 10^{-7}$ (as shown in Figure 4.14).

The simulations were conducted with 10 sensor nodes, 8 voltage levels, 3 modulation levels ($\{2, 4, 6\}$), 60 computation tasks, $u \in \{0.0, 0.2, 0.5, 0.8, 1.0\}$, and $CCR \in [0, 20]$. Compared with Figure 4.11, we observe a significant amount of performance improvement in Figure 4.15. For example, when $u = 0.5$, the highest lifetime improvement increased from 3x in Figure 4.11(a) to 6x in Figure 4.15(a), and even 10x in Figure 4.15(b). The difference in performance improvement between Figures 4.15(a) and 4.15(b) is because a larger C_{tr} led to a larger energy/time ratio of communication activities. This, in turn, gave more advantage in reducing the communication energy by utilizing modulation scaling.

Similar to Figure 4.11, larger improvements were observed when u be-

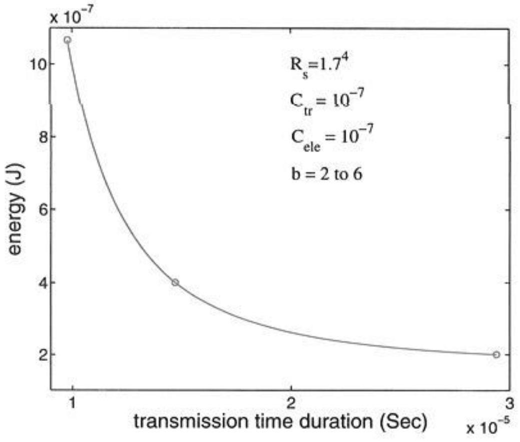


Fig. 4.14 Energy *vs* latency tradeoffs for transmitting one bit of data.

came smaller. In addition, the miss rate of the heuristic exhibited a similar trend as in the cases with only voltage scaling.

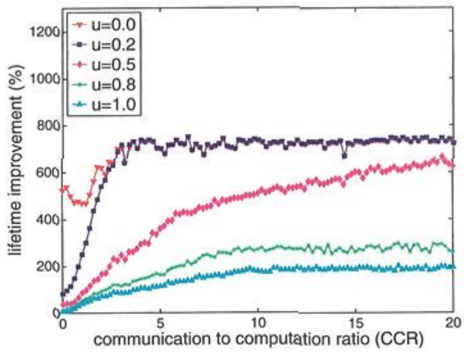
4.6.2 *Application Graphs from Real World Problems*

In addition to synthetic application graphs, we also considered application graphs of two real world problems: the LU factorization algorithm [42] and the Fast Fourier Transformation [44]. These two algorithms are widely used as kernel operations for various signal processing applications, such as beamforming [127].

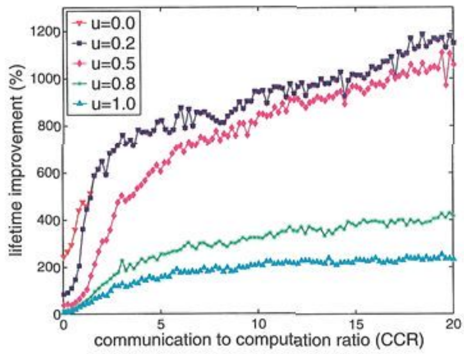
4.6.2.1 *LU Factorization*

Figure 4.16(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 4.16(b). Each $C_{k,k}$ represents a pivot column operation and each $C_{k,j}$ represents an update operation. The total number of computation 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.

We performed simulations with 10 sensor nodes, 8 voltage levels, 4 channels, 3 modulation levels, $C_{tr} = 10^{-6}$, and the matrix dimension, s, varying from 5 to 20. It is easy to verify that the computation requirement of any task, $C_{k,j}$, was $s - k$ ALU operations. Further, for any task, $C_{k,j}$, the size



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



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

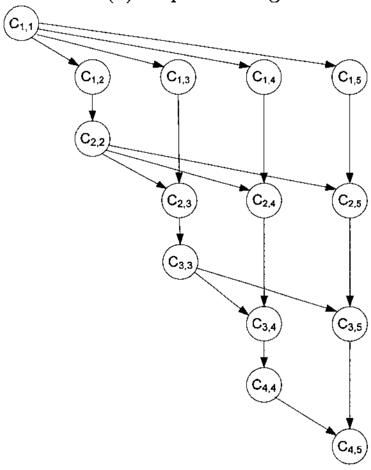
Fig. 4.15 Lifetime improvement of the 3-phase heuristic, with modulation scaling (10 sensor nodes, 8 voltage levels, 4 channels, 3 modulation levels, 60 computation tasks).

of data transmitted by any communication activity to the task was $s - k$ 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 4.17. The performance of the heuristic improved when CCR increased or u decreased. The lifetime im-

```
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



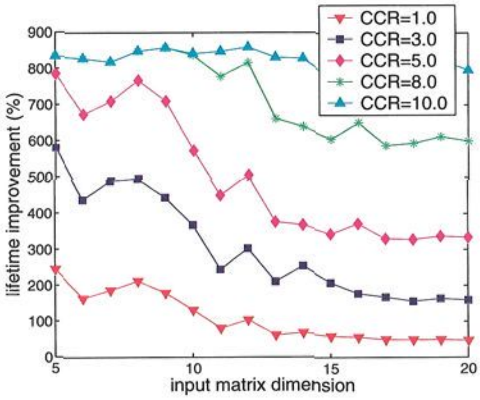
(b) Example application graph with a 4×4 matrix

Fig. 4.16 Matrix factorization algorithm.

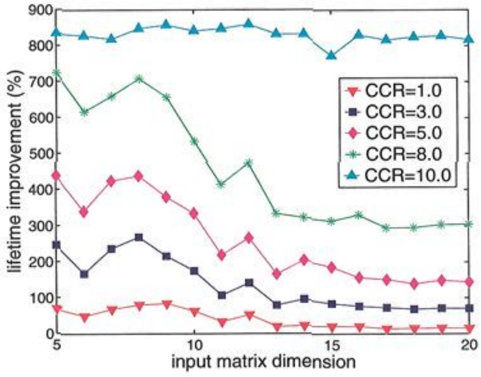
provement approached 8x when $CCR = 10.0$. Also, very little improvement was observed during our simulations when setting CCR beyond 10.0. The least amount of lifetime improvement was about 15%, when $u = 0.8$, $CCR = 1.0$, and $s = 20$.

4.6.2.2 *Fast Fourier Transformation (FFT)*

The recursive, one-dimensional FFT Algorithm is given in Figure 4.18(a). In the figure, A is an array of length l, which holds the coefficients 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



(a) $u = 0.5$



(b) $u = 0.8$

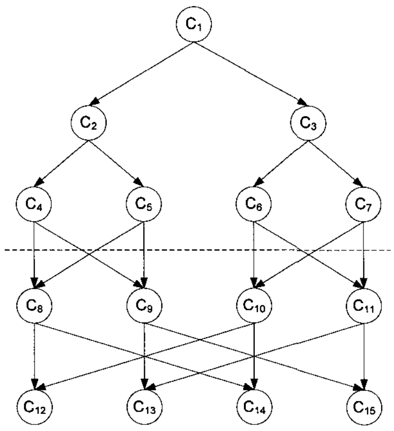
Fig. 4.17 Lifetime improvement for the matrix factorization algorithm (10 sensor nodes, 8 voltage levels, 4 channels, 3 modulation levels).

tasks and $l \times \log l$ butterfly operation tasks (we shall assume $l = 2^k$ for some integer k). For example, the application graph with four data points is given in Figure 4.18(b) . The seven tasks above the dashed line are the recursive call tasks, while the eight tasks below the line are butterfly operation tasks.

We performed simulations using 10 sensor nodes, 8 voltage levels, 4

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

(a) Sequential algorithm

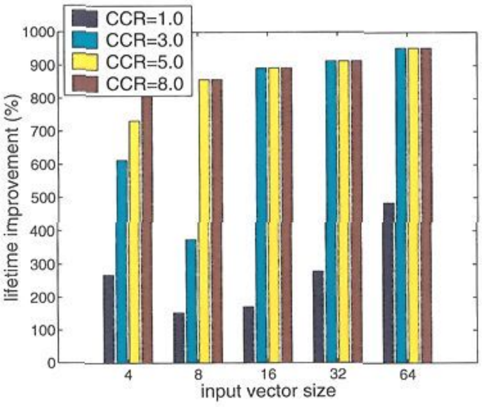


(b) Example application graph with 4 points

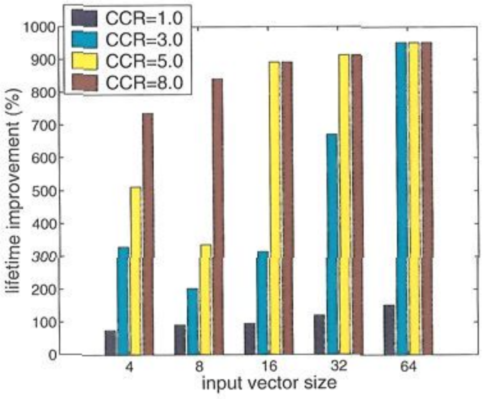
Fig. 4.18 Fast Fourier Transformation (FFT) algorithm.

channels, 3 modulation levels, and $C_{tr} = 10^{-6}$. The vector size was varied from 4 to 64 incrementing by powers 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 4.19. Again, the performance of the heuristic improved when CCR increased or u decreased. The lifetime improvement was close to 10x when $CCR = 8.0$ and $l = 64$. The least amount of lifetime improvement was around 75%, when $u = 0.8$, $CCR = 1.0$, and $l = 4$.



(a) $u = 0.5$



(b) $u = 0.8$

Fig. 4.19 Lifetime improvement for the FFT algorithm (10 sensor nodes, 8 voltage levels, 4 channels, 3 modulation levels).

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 cases where data are locally sensed or gathered at each individual sensor node. For example, in Figure 4.18(b), input data can be generated by tasks

C_4 to C_7 through local sensing. Thus, the recursive calls above the dashed line to disseminate the data become unnecessary.

4.7 Summary

In this chapter, we have investigated the problem of allocating a periodic real-time application to a single-hop cluster of homogeneous sensor nodes with multiple wireless channels. The key technique was to explore the tradeoffs between energy *vs* latency for both computation and communication tasks, via voltage and modulation scaling. A new performance metric has been proposed to balance the energy dissipation among all the sensor nodes. We have presented both an ILP formulation and a polynomial time heuristic.

We have demonstrated, using simulations, that for small scale problems, a lifetime improvement of up to 5x was achieved by the ILP-based approach, compared to the case where no voltage scaling is used. Also, the performance of the 3-phase heuristic achieved up to 63% of the system lifetime obtained by the ILP-based approach. For large scale problems, a lifetime improvements of up to 10x was observed when both voltage and modulation scaling were used. Simulations were also conducted for application graphs from LU factorization and FFT. The 3-phase heuristic achieved a lifetime improvement of up to 8x for the LU factorization algorithm, and an improvement of up to 9x for the FFT algorithm.

This page is intentionally left blank

Chapter 5

Information Transportation over a Tree Substrate

5.1 Overview

5.1.1 *Motivation*

In the last chapter, we showed the application of voltage scaling and rate adaptation for information processing within a cluster of collocated 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. As stated in Chapter 2, 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 — the 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, thus reducing the communication load.

For applications with small volumes of data and simple aggregation operations, communication is a significant source of energy dissipation in the process of information routing. It is therefore important to design energy-efficient communication strategies. As we pointed out in Chapter 2, rate adaptation is an important technique for improving the energy efficiency of communication. Therefore, in this chapter, we study the problem of scheduling packet transmissions over a data gathering tree, using rate adaptation.

We consider a real-time mission-critical scenario, where the raw data gathered from the source nodes must be aggregated and transmitted to the sink within a specified latency constraint. Our objective is to minimize the overall energy cost of the sensor nodes in the data gathering tree, subject to the latency constraint. Although our problem is formulated as a convex programming problem, which is solvable in polynomial time by using general optimization tools, we propose a more time-efficient algorithm in this chapter, which exploits special properties of the problem. Such properties include the convexity of the energy function of wireless communication and the tree structure of the underlying communication substrate.

It is important to evaluate the usefulness of the latency-energy tradeoffs by examining the sources of latency and energy costs for data gathering in WSNs. We assume a low-duty cycle WSN with sleep scheduling, so that nodes are completely shut down in their idle state. In such a system, besides the time cost for packet transmission, the latency for data gathering can be further decomposed into queuing delay, channel access delay, re-transmission delay, and the delay for waking up sleeping nodes. We believe that under several reasonable assumptions, the packet transmission delay is significant and should be traded for energy efficiency.

First, due to the application-specific design of WSNs, most traffic throughout the network is to transport the gathered data to the base station. It is also anticipated that many applications for WSNs require the transmission of tens to hundreds of bytes per second [191]. In such a light-traffic scenario, queuing delay is not such a major concern as it is in traditional wireless ad hoc networks. Second, we assume the availability of multi-packet reception (MPR) techniques [176; 186], so that channel access delay due to collision detection and avoidance is negligible. Third, the number of expected re-transmissions is actually a function of the Bit Error Rate at the receiving node, which in turn determines the tradeoffs of energy *vs* latency for packet transmission (see Section 3.2.2 for details). Thus, it is convenient to incorporate explicitly the tradeoffs between the expected number of re-transmissions and energy into our work. Fourth, we assume the availability of an ultra-low power wakeup radio [215; 161], which has been discussed in Chapter 4. This wakeup radio can be used to synchronize packet transmission between sensor nodes with almost no delay or energy penalties. Also, the typical startup time for sensor nodes is around 100 μSec [191], while the time for transmitting a packet of 200 bytes using 1 Mbps is 200 μSec. Based on the above observations, the time for packet transmission in light-traffic WSN applications constitutes a significant portion of the overall delay.

Since we assume that sensor nodes are completely shut down in idle state, the main source of energy cost is due to packet transmission for data gathering. It is therefore crucial to explore the tradeoffs between the energy and latency of packet transmission in such a context.

5.1.2 *Technical Overview*

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*. Based on this numerical optimization technique, we are able to analyze the energy gain of our approach for a special case with a complete binary data gathering tree.

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 identify iteratively the sensor node with the highest potential energy reduction in the gathering tree (to be explained later), and reduce its energy cost when allowed by the latency constraint. In this protocol, each sensor node only needs to perform simple operations 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 extensively with simulations, for both long and short-range communications described in Chapter 3. We consider two models of source placement, namely the random source and the event radius source placements [100]. We use the baseline where all sensor nodes transmit the packets at the highest speed (8 bits/symbol), and shut down their radios afterward. Our simulation results from the studied scenarios show that, compared to 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. Furthermore, the adaptability of the protocol is demonstrated with two run-time scenarios.

5.1.3 *Chapter Organization*

We briefly discuss the related work in Section 5.2. We describe our underlying network model in Section 5.3. The packet transmission problem is then defined in Section 5.4. Off-line algorithms for the problem are presented in Section 5.5. In Section 5.6, a distributed on-line protocol is described. Simulation results are shown in Section 5.7. The chapter is summarized in Section 5.8.

5.2 Related work

The data gathering tree is common to data-centric information routing in WSNs [85; 100]. The construction of the data gathering tree has been studied under various circumstances, as previously summarized in Section 2.4.4. For example, several localized tree topology generation mechanisms are compared by Zhou *et. al.* using metrics including node degree, robustness, and latency [216]. A randomized logarithmic approximation algorithm is developed by Goel *et. al.*, for the case when the joint entropy of multiple information sources is modeled as a concave function of the number of sources [67]. By considering a simplified compression model, where the entropy conditioning at nodes depends only 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 [47]. An analysis of the impact of spatial correlation on several practical schemes for tree construction [135] indicates that a simple cluster-based routing scheme performs well, regardless of 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 a wireless communication perspective, rate adaptation has been widely studied to optimize spectral efficiency (e.g., network throughput), subject to the channel conditions in cellular networks [5; 15; 188], or local-area wireless networks [6; 77; 211]. Several recent works [64; 143; 146; 157; 158; 209] have studied the application of rate adaptation to 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.* [143]. An extension of the problem [64] investigates the packet transmission from a single transmitter to multiple receivers. In both [143] and [64], optimal off-line algorithms and near-optimal on-line solutions are provided. The concept of modulation scaling was first proposed by Schurgers *et. al.* [157]. For a single-hop link, policies for adjusting the modulation level are developed for cases where no real-time requirements are imposed [157], or each packet delivery has a deadline to meet [158]. Moreover, modulation adaptation is integrated into multi-packet scheduling with deadlines for each packet [158], and also the Weighted Fair Queuing (WFQ) scheduling policy [146]. For a multi-hop communication path, modulation scaling is used to balance the energy cost of all nodes along the path [209].

Similar to the work presented in Chapter 4, the real-time latency constraint considered in this chapter requires the use of global time-synchronization schemes [57]. Our scenario is similar to the epoch-based data gathering scheme [115], 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.

The challenges of our problem are multi-fold. Firstly, the energy functions can vary for different links. It is therefore necessary 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 apply the techniques in [64] and [143] directly, as they require explicit latency constraints over each link. Lastly, as described in Section 3.2.2, 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 works. Albeit the above challenges, the tree structure leads us to an extension of the numerical optimization algorithm proposed in [64] as well as a recursive representation of the problem to apply dynamic programming.

5.3 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 in Table 5.1 a summary of notations used in this chapter.

Table 5.1 List of notations.

$T =< \mathbb{V}, \mathbb{L} >$	The data gathering tree composed by the set of v sensor nodes $\mathbb{V}$ and the set of communication links $\mathbb{L}$
$\{V_1, \ldots, V_M\}$	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

5.3.1 *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 en route 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 sink, we use a directed graph representation. Let $T =< \mathbb{V}, \mathbb{L} >$ denote the data gathering tree, where $\mathbb{V} = \{V_i : i \in [v]\}$ denotes the set of v sensor nodes and $\mathbb{L} = \{L_i : i \in [v-1]\}$ denotes the set of directed communication links between the sensor nodes. 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 $\mathbb{V}$ (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.

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.

We assume a simplified communication model with a medium access control (MAC) layer that ensures no collision or interference at a node, which can be realized by multi-packet reception (MPR) techniques through frequency, code, or spatial diversity [176; 186]. We also assume that sensor nodes are completely shut down in the idle state, and can be awakened for packet transmission using an ultra-low power wakeup radio [215; 161], with almost no delay or energy penalties. The sensing and computation cost for data aggregation are considered to be negligible.

5.3.2 *Data Aggregation Paradigm*

Various techniques have previously been proposed for computing aggregates, or joint information entropy, from multiple source nodes. In our study, we adopt the model proposed by Pattern *et. al.* [135], 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 factor 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 [135]:

$$H_2 = H_1 + \frac{d}{d+c} H_1 \tag{5.1}$$

We assume that the correlation parameter c is the same for any set of sources. Based on (5.1), a recursive calculation of the total compressed information of multiple sources can be developed [135]. We omit the details here. (Interested readers may refer to [135].)

Although we use (5.1) as a typical aggregation function, our technique is not limited to this function alone. The only requirement is that we can derive the values of s_i, for all i, based on the functions. Thus, even different functions can be used to specify the aggregation at different sensor nodes.

5.4 Problem Definition

A schedule of packet transmission is defined as a vector $\vec{\tau} = \{\tau_i : i \in [v-1]\}$, where τ_i is the time duration of 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 of packet transmission of all the sensor nodes in the data gathering tree.

We use the energy model described in Section 3.2.2. We re-write the equations for determining the time and energy costs of transmitting a packet of size s:

$$\tau = \frac{s}{b \cdot R_s} , \tag{5.2}$$

$$w(\tau) = [C_{tr} \cdot (2^{\frac{s}{\tau \cdot R_s}} - 1) + C_{ele}] \cdot \tau \cdot R_s , \tag{5.3}$$

where b is the modulation level, R_s is the symbol rate, 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.

Let $w_i(\tau_i)$ denote the energy function of V_i with potentially variable values of parameters C_{tr}, C_{ele}, and R_s for different links. Let m_i denote 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):

Given:

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

b. energy functions for each link $(i,j) \in \mathbb{L}$, $w_i(\tau_i)$,

c. the latency constraint, Γ;

find *a schedule of packet transmission, $\vec{\tau} = \{\tau_i : i \in [v-1]\}$, that minimizes*

$$f(\vec{\tau}) = \sum_{i=1}^{n-1} w_i(\tau_i) \tag{5.4}$$

subject to

$$\forall p_i \text{ in } T, \Phi_i = \sum_{j:V_j \in p_i} \tau_j \leq \Gamma \, . \tag{5.5}$$

We consider both an off-line version and an on-line version of PTP. 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 has only local knowledge of its own radio status, and can communicate with its parent and children. Thus, distributed on-line protocols are needed to adapt the transmission time of each sensor node locally, to achieve global energy minimization.

5.5 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 [64] to get optimal solutions for OPTP. A faster dynamic programming based approximation algorithm is then presented. Techniques for handling interference are also discussed.

5.5.1 *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 5.1 *A schedule, $\vec{\tau}^*$, is optimal for OPTP if and only if*

(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

$$\dot{w}_i(\tau_i^*) = \sum_{(j,i)\in\mathbb{L}} \dot{w}_j(\tau_j^*)\,. \tag{5.6}$$

The proof of the lemma is presented in Appendix A.

The problem proposed in [64] 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 proposed 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 [64] can be directly applied to solve such a special case.

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 5.1. 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 of the links is set to 0 (Steps 2 and 3). The main idea is to increase (move right) iteratively the starting times of packet transmissions, so that each move locally optimizes our objective function. 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 5.1 holds for the subtree formed by V_i and its parent and children, with respect to the invariant that $\tau_j^k \le m_j$ for any node V_j in the subtree. When the best($\cdot$) function is called upon the subtree around V_i, the transmission for all the links not within the subtree remains fixed. That is to say, the starting time of transmissions from the children of V_i and the ending time of the transmission from V_i are fixed. We can prove that the starting time of the transmission from V_i will never be decreased by calling the best($\cdot$) function (refer to Appendix A). Hence, in the best($\cdot$) function, the locally optimal starting time of the transmission from V_i is obtained by a binary search between the original starting time and the

```
Begin
1. $k \leftarrow 0$                                      // initialize iteration counter
2. For $(i, v) \in \mathbb{L}$                           // initialize transmission time
3.     $\tau_i^k \leftarrow \min\{\Gamma, m_i\}$         // for links to the sink
4. For $(i, j) \in \mathbb{L}$ such that $j \neq n$      // initialize transmission time for
5.     $\tau_i^k \leftarrow 0$                           // other links
6. $flag \leftarrow 0$                                   // flag for convergence
7. While $flag = 0$
8.     $k \leftarrow k + 1$                              // increment the iteration counter
9.     For each $V_i$ with $i$ from $v - 1$ downto M+1   // local optimization
                                                         // for internal nodes
10.        $(\{\tau_j^k\}_{(j,i)\in\mathbb{L}}, \tau_i^k) \leftarrow$ best$(\{\tau_j^{k-1}\}, \tau_i^{k-1})$   // move right the start
                                                         // time of transmission
                                                         // from $V_i$
11.    For $(i, v) \in \mathbb{L}$
12.        $\tau_i^k \leftarrow \min\{m_i, \Gamma - (\max_{V_i \in p_j}\{L_j\} - \tau_i^k)\}$   // increase the
                                                         // transmission time
                                                         // for links
                                                         // to the sink
13.    If $\vec{\tau}^k = \vec{\tau}^{k-1}$, $flag \leftarrow 1$   // check convergence
End
```

Fig. 5.1 Pseudo code for EMR-Algo.

ending time of the transmission. Step 10 moves right the complete time of transmissions on links to the sink. This movement stops when the latency constraint is reached.

The correctness of EMR-Algo can be proved by exploring the convexity property of the energy functions. Let $\vec{\tau}^* = \{\tau_1^*, \dots, \tau_{v-1}^*\}$ be the optimal schedule. Let $\theta_i^* = 0$, for $i = 1, \dots, M$; and $\theta_i^* = \max_{(j,i)\in\mathbb{L}}(\theta_j^* + \tau_j^*)$, for $i = M, \dots, v-1$. As previously stated, $\{\tau_i^k : k = 1, \dots, v-1\}$ indicate the transmission time of nodes $V_1, \dots, V_{v-1}$ after the k-th pass of EMR-Algo. Let $\theta_i^k = 0$, for $i = 1, \dots, M$, and $\theta_i^k = \max_{(j,i)\in\mathbb{L}}(\theta_j^k + \tau_j^k)$, for $i = M, \dots, v-1$.

Theorem 5.1 *Let θ_i^k and θ_i^*, $i = 1, \dots, n-1$ be defined as before. Then*

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

The proof of Theorem 5.1 is detailed in Appendix A.

5.5.2 *Performance Analysis for a Special Case*

We consider a special case where the data gathering tree is a complete binary tree, where all the leaf nodes are source nodes. Let d denote the depth of the tree, with 2^i nodes at depth $i \leq d$. Let s denote the size of data packet from all sources. We assume a perfect data aggregation at all nodes, such that the packets on all edges are of size s.

For simplicity, we assume that the communication environments and radio devices for all nodes are identical in terms of parameters R_s and C_{tr}. We assume a long-range communication scenario where F is negligible. Due to the structure of a complete binary tree, it can be inferred that nodes at the same depth shall have the same transmission duration in the optimal solution. Let τ_i denote the transmission time of nodes at depth i, where $1 \leq i \leq d$.

Based on (5.3), the PTP problem in this special case (denoted PTP-SP), is to find a schedule $\{\tau_i : i = 1, \ldots, d\}$ in order to:

$$\text{minimize} \sum_{i=1}^{d} 2^i [C_{tr}(2^{\frac{s}{\tau_i R_s}} - 1) \cdot \tau_i \cdot R_s] \tag{5.7}$$

$$\text{subject to} \sum_{i=1}^{d} \tau_i \leq \Gamma \,. \tag{5.8}$$

As $\Gamma \to \infty$, the lower bound on the cost of PTP-SP approaches $(2^{d+1} - 2)sC_{tr} \ln 2$. Thus, we set Γ in the range $[\frac{ds}{8R_s}, \frac{ds}{2R_s}]$ to avoid this trivial case. The boundaries of the range are obtained by setting the modulation level of all sensor nodes to 8 and 2, respectively. We also consider the baseline, where all sensor nodes transmit with a modulation level of 8 and shut down afterward, i.e., $\tau_i = \frac{s}{8R_s}$ for all $i \leq d$. The cost of such a baseline is $(2^{d+1} - 2)\frac{255 s C_{tr}}{8}$, which is an upper bound on the cost of PTP-SP.

Based on Lemma 5.1, it can be shown that the optimal schedule to PTP-SP satisfies

$$2^i \cdot C_{tr} \cdot R_s (2^{\frac{s}{\tau_i R_s}} - 1 - 2^{\frac{s}{\tau_i R_s}} \cdot \ln 2 \cdot \frac{s}{\tau_i R_s}) = \lambda \tag{5.9}$$

where λ is Lagrange multiplier determined by the constraint $\sum_{i=1}^{d} \tau_i = \Gamma$. Let $b_i = \frac{s}{\tau_i R_s}$ be the modulation level for nodes at depth i. Since $b_i \geq 2$, we can approximate (5.9) as $2^{b_i} = \frac{\lambda}{2^i C_{tr} R_s}$. This approximation is similar to the inverse-log scheduling in [205], which is shown to be a close approximation for (5.9) when b_i is large. This approximation leads to $\tau_i = \frac{s}{R_s \log \frac{\lambda}{2^i C_{tr} R_s}}$, where λ is determined by $\sum_{i=1}^{d} \tau_i = \Gamma$.

To solve λ, let $x = \log \frac{\lambda}{C_{tr} R_s}$ and $\kappa = \frac{\Gamma R_s}{s}$. The constraint $\sum_{i=1}^{d} \tau_i = \Gamma$ can be written as

$$\sum_{i=1}^{d} \frac{1}{x-i} = \kappa \,, \tag{5.10}$$

which is essentially a polynomial equation in x of degree d. Since $\tau_i > 0$ for all $i \leq d$, we consider the only root that satisfies $x > d$. Let $Harm(i) = 1 + \frac{1}{2} + \ldots + \frac{1}{i}$ denote the Harmonic function. We have $\sum_{i=1}^{d} \frac{1}{x-i} \approx Harm(x-1) - Harm(x-d-1)$. We use the approximation $Harm(i) = \ln(i + \frac{1}{2}) + \gamma$ for large i, where γ is the Euler-Mascheroni constant. Thus, we derive

$$x \approx \frac{(2d+1)e^{\kappa} - 1}{2(e^{\kappa} - 1)} \,. \tag{5.11}$$

When d is large, we have an approximated optimal schedule:

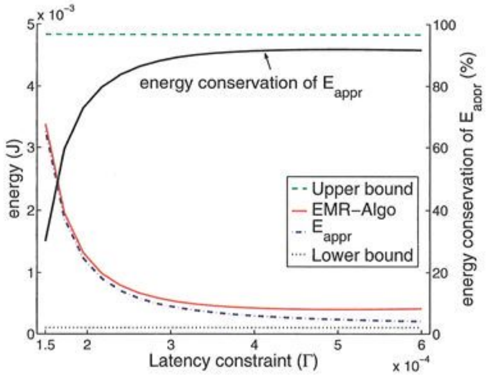
$$\tau_i = \frac{s}{R_s(x-i)} \tag{5.12}$$

for $i = 1, \ldots, d$, with an energy cost C_{appr}:

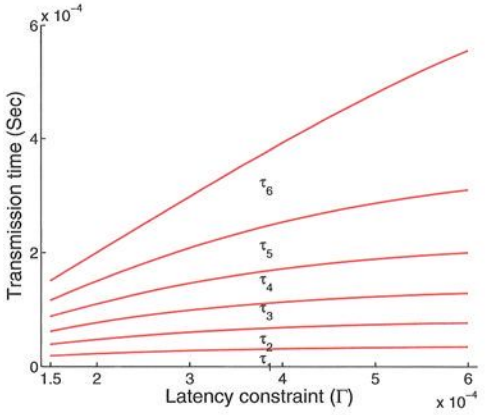
$$\begin{aligned} C_{appr} &= \sum_{i=1}^{d} 2^i C_{tr} (2^{\frac{s}{\tau_i R_s}} - 1) \tau_i R_s \\ &= s C_{tr} \sum_{i=1}^{d} \frac{2^i (2^{x-i} - 1)}{x-i} \\ &\approx s C_{tr} 2^x \sum_{i=1}^{d} \frac{1}{x-i} && \text{(when } d \text{ is large)} \\ &= s C_{tr} 2^x \kappa && \text{(from (5.10))} \,, \end{aligned} \tag{5.13}$$

where $\kappa = \frac{\Gamma R_s}{s}$ and x is given by (5.11). This gives an improvement over the baseline by a factor of approximately $\frac{2^{d+6-x}}{\kappa}$.

In Figure 5.2(a), with $C_{tr} = 6 \times 10^{-9}$, $R_s = 10^6$, $s = 200$, and $d = 6$, we plot C_{appr} and the cost obtained by EMR-Algo as a function of Γ as well as the lower and upper bounds on the cost of PTP-SP. We observe that when Γ is small, C_{appr} and the cost of EMR-Algo are nearly equal. When Γ is large, although there is noticeable difference between C_{appr} and the



(a) Comparison between E_{appr} and the cost of EMR-Algo



(b) Sensor node transmission time

Fig. 5.2 Performance analysis for a special case over a complete binary data gathering tree.

cost of EMR-Algo, the ratios of their improvement over the upper bound are actually quite similar. Notice that when Γ is at the minimum, there is still improvement by C_{appr} and EMR-Algo over the upper bound. This is because, for C_{appr} and EMR-Algo, we have relaxed the constraint that b_i must lie within $[2, 8]$. We will show in Section 5.7.3 that for our on-line protocol, which considers such a constraint, the resulting energy savings are still comparable to that of EMR-Algo.

We also plot the energy conservation of E_{appr}, which is defined as the percentage of energy savings by E_{appr} over the upper bound. We observe an energy conservation from 30% to 90% for E_{appr}. Although these numbers are based on the above special case, they confirm our simulation results for general trees in Section 5.7 very well. Our analysis on this special case gives meaningful insight into the energy conservation that can be achieved by our technique in general scenarios.

In Figure 5.2(b), we also plot τ_i with respect to variations in Γ given by (5.12). It is observed that when Γ increases, the transmission time of nodes with larger depth increases faster than that of nodes with smaller depth. This is because the number of nodes increases exponentially with depth. Thus, more transmission time is desired for nodes with large depth, to sustain Lemma 5.1.

5.5.3 *A Dynamic Programming-Based Approximation Algorithm*

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 5.7, we show the EMR-Algo running time 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, which is presented in this section.

For ease of analysis, we assume that for each sensor node, D discrete values of τ are evenly distributed over $[0, \Gamma]$. 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. 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 $g(V_i, t)$:

$$g(V_i, t) = \begin{cases} w_i(t), \text{ for } 1 \leq i \leq M & (5.14a) \\ \sum_{(k,i)\in\mathbb{L}} (\min_{j=1}^{\frac{t}{\varepsilon}}\{w_k(j\varepsilon) + g(V_k, t - j\varepsilon)\}), \text{ otherwise} & (5.14b) \end{cases}$$

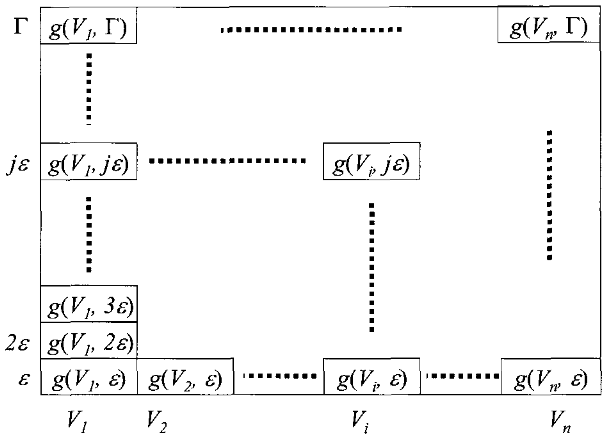


Fig. 5.3 The $g(\cdot)$ table computed by DP-Algo.

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 5.3). The i-th column from the left side corresponds to sensor node V_i, while the j-th row from bottom corresponds to $j\varepsilon$. In the output of the DP-Algo, the cell crossed by the j-th row and the i-th column contains the value of $g(V_i, j\varepsilon)$.

To build the table, we start from the bottom-left cell, which 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 5.2, it can be verified that the values of τ_i resulting from different modulation levels are not evenly distributed over $[0, \Gamma]$. Thus, DP-Algo cannot be directly applied. However, one practical method is, for each i, to set τ_i obtained by EMR-Algo or DP-Algo to the largest time duration below τ_i that can be achieved by an available modulation level. We call this method the *rounding procedure.* The rounding procedure may affect the performance of DP-Algo. As shown in Section 5.7, the performance degradation is around 10% for loose latency constraints, and 50% for stringent latency constraints.

Another issue with modulation scaling is that the maximum or minimum transmission times can be bounded, due to the lowest and highest settings for the modulation level. Such boundaries can be achieved by assigning infinity to the corresponding cells in the table.

5.6 A Distributed On-Line Protocol

The algorithms presented in Section 5.5 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 only local information from sensor nodes.

We define the *energy gradient* of a sensor node as the amount of energy that can be saved by increasing its transmission time to the next level. The key idea of the protocol is to identify iteratively the sensor nodes with the largest positive energy gradient, and increase their transmission time if the latency constraint allows. 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 at V_i.
(3) Every sensor node can measure its current power consumption, and therefore its energy gradient – the energy gain by increasing the transmission time of the node by ε.
(4) Interference among sensor nodes is handled with MPR techniques.

The local identifier in Assumption 1 is commonly implemented in protocols such as Directed Diffusion [85]. Assumption 2 can be fulfilled by attaching a time stamp to each packet from the leaf nodes (we shall assume that time synchronization schemes, such as [57], 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 (e.g., 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, so as to resolve collisions.

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.

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 of its children, the node compares the energy gradients piggybacked to each packet to 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. This procedure continues until all the sensor nodes have the correct values of r. Fields τ_i and τ_d can be easily maintained based on Assumption 2.

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 the latency laxity of nodes along this longest path. Node V_i then takes one of the following actions.

(1) If $\delta < 0$, the transmission time for a 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) 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 this 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. This 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 $V_i \in \mathbb{V}$. We assume that each sensor node has q discretized transmission times. Before the protocol converges, a feedback packet increases 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 when implementing the protocol. Ideally, the adaptation should be performed in a stable system state. Thus, the period α for disseminating the feedback packet should be large enough to accommodate oscillations in the 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 is lost 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 reduce repeatedly the transmission time of the sensor nodes with the smallest energy gradient, until the latency constraint is satisfied. This option is more aggressive than the proposed protocol, in the sense of reducing the incurred increment in energy cost. However, it requires more sophisticated control protocol, and more importantly, increases the response time in handling latency violations.

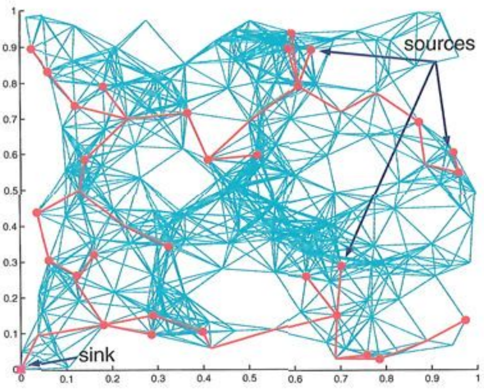
The above protocol actually does not require the discretized transmission time to be evenly distributed with distance ε. In the example of modulation scaling, the set of transmission times is generated based on $\frac{s}{b \cdot R_s}$, by varying b within $[2, 4, 6, ...]$. The distance between adjacent transmission times decreases with b. This can be handled by the following modification to the protocol. First, in Action 1, after decreasing by a factor of β, the transmission time is rounded down 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 in Section 5.7.

5.7 Simulation Results

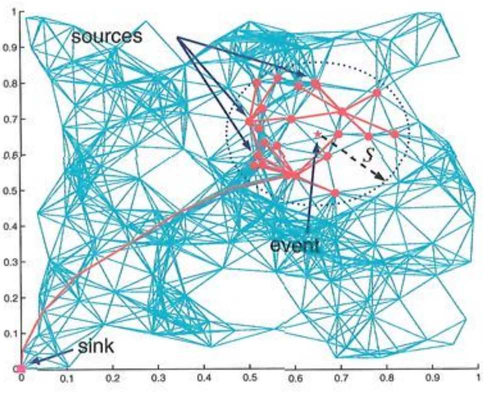
To conduct the tests, a simulator was developed using the PARSEC [134] software, which is a discrete-event simulation language. The purposes of the simulations were (1) to demonstrate the energy gain achieved by our algorithms compared to the baseline; (2) to evaluate the impact of several key system parameters to the performance of our algorithms; and (3) to evaluate the energy savings and the adaptation capability of our on-line protocol in various run-time scenarios.

5.7.1 *Simulation Setup*

The transmission speed of sensor nodes was continuously tunable by setting the modulation level within $[2, 8]$, except for the special case of modulation scaling where the modulation level could only be even integers between 2 and 8. Hence, for all sensor nodes, the highest data rate was 8 Mbps and the lowest data rate was 2 Mbps. The baseline in our simulations was to transmit all packets at the highest speed (i.e., 8 Mbps), and shutdown the radios afterward. This policy is used, for example, in the PAMAS [145] and the D-MAC protocols [109]. In our simulations, the performance metric was defined as the percentage of energy savings achieved by using our techniques, compared to the baseline.



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



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

Fig. 5.4 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 bottom-left corner of the square. The neighbors with which a sensor node could directly communicate were determined by a connectivity parameter, $\rho \in (0, 1]$. Specifically, two sensor nodes could communicate with each other only if the distance between them was within ρ. Note that ρ was a purely relative measurement in the unit square. Consider the example when $\rho = 0.1$. In the case of short-range communication, ρ was translated to 7 m when the square scaled to 70 m $\times$ 70 m. In the case of long-range communication, ρ was translated to 30 m when the square scaled to 300 m $\times$ 300 m. The size of raw data from all source nodes was set to 200 bits.

We used two models to generate 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 were randomly selected to be the sources, whereas in the ER model, all sources were located within a distance S (essentially the sensing range) of a randomly chosen "event" location. For both models, the Greedy Incremental Tree (GIT) algorithm [100] was used for constructing the data gathering tree. In Figure 5.4, we illustrate two example data gathering trees generated based on the RS and ER models.

The energy function used in the simulation was in the form of (5.3). Unless otherwise stated, $R_s = 10^6$ and $C_{ele} = 10^{-8}$ for all the sensor nodes, while the value of C_{tr} of a sensor node was determined by the distance from the node to its parent in the tree. Specifically, we assumed a d^2 power loss model, where d was the distance between a node and its parent. Then, for node V_i, we had its $C_{tr} = C_{base} \cdot (\frac{d}{\rho})^2$. Based on our analysis in Section 3.2.2, 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 constraint 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 $\min\{m_i, \frac{s_i}{2}\}$. 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}.

The presented data is averaged over more than 150 problem instances, and has a 95% confidence interval with a 10% (or better) precision.

5.7.2 *Performance of the Off-Line Algorithms*

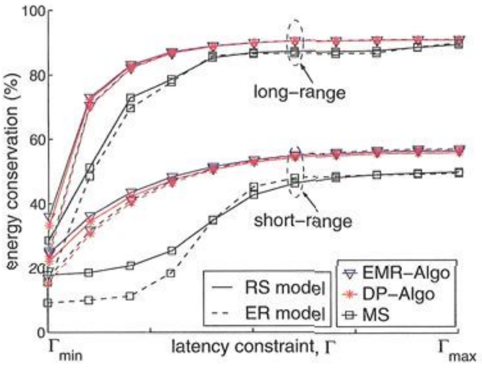


Fig. 5.5 Performance of our off-line algorithms (c: correlation parameter, ρ: connectivity parameter, N: number of sources, S: sensing range).

5.7.2.1 *Performance Overview*

In Figure 5.5, we show 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 approximation accuracy for DP-Algo, D was set to 100.

As Γ approached Γ_{max}, our algorithms achieved more than 90% energy savings for the long-range communication, and around 50% for the short-range communication. When $\Gamma = \Gamma_{min}$, EMR-Algo and DP-Algo could still save more than 30% of the energy for long-range communication, and 20% for short-range communication.

The reason for successful energy savings when $\Gamma = \Gamma_{min}$ is because Γ equals the transmission time of the longest path in the data gathering tree. Thus, energy can still be 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 to the performance in the RS model. Specifically, as illustrated in Figure 5.4, 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 like a star structure.

The plot shows that the performance of DP-Algo was quite similar to the performance of EMR-Algo. However, the performance of MS quickly degraded as Γ decreases. From Figure 3.3, the first derivative of the energy function decreased quickly as τ approached 0. Thus, when Γ was small, the rounding procedure for solutions with high modulation levels led to large performance loss.

During our simulation, we also observed that when $\Gamma = \Gamma_{min}$, MS failed 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 (the miss rate), is given in Table 5.2.

Table 5.2 The miss rate of MS (based on simulated instances for Figure 5.5).

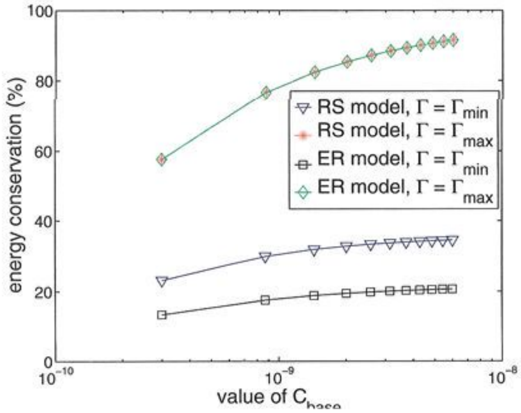
Source model	Communication scenario	# of successful instances	# of failed instances	Total number of instances	Miss rate (%)
RS	Long-distance	102	28	130	21
	Short-distance	104	16	120	13
ER	Long-distance	324	66	390	17
	Short-distance	352	38	390	10

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

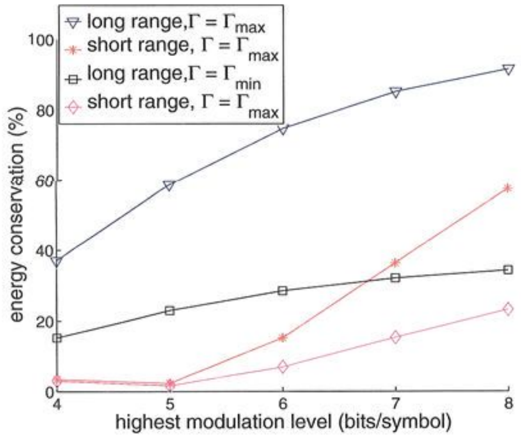
5.7.2.2 *Impact of Radio Parameters*

We studied the impact of radio parameters, including C_{base}, R_s, the highest modulation level, and the start-up energy. In Figure 5.6(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 increased with C_{base}. Also, when $\Gamma = \Gamma_{max}$, there was 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% was observed for the ER model compared to the RS model.

To evaluate the impact of symbol rate R_s, we varied R_s from 10 KBaud



(a) Impact of C_{base}



(b) Impact of the highest modulation level

Fig. 5.6 Impact of radio parameters (correlation parameter $c = 0.5$, connectivity parameter $\rho = 0.15$, number of sources $N = 30$, sensing range $S = 0.2$).

to 1 MBaud. Consider the modulation level within $[2, 8]$, the above range of R_s reflected a bit rate of 20 to 80 Kbps when $R_s = 10$ KBaud, and 2 to 8 Mbps when $R_s = 1$ MBaud. We observed that the energy savings were almost the same throughout the variation of R_s. This was understandable since, from (5.3), the performance ratio of DP-Algo to the baseline is determined by b, not R_s.

We further investigated the performance of DP-Algo under different settings of the highest modulation level of the radio. In Figure 5.6(b), we show the energy savings achieved by DP-Algo in RS model when the highest modulation level was varied from 4 to 8. As expected, a lower highest modulation level resulted in less energy savings. When the modulation level was restricted to [2, 4], less than 20% energy savings were achieved for both long and short range communication.

We have also studied the impact of radio start-up energy, which was estimated to be 1 μJ [147]. In each epoch, the radio of each sensor node was started exactly once. Our simulation results showed that the impact of the start-up energy to the long-range communication was almost negligible. However, compared to Figure 5.5, we observed a decrease of 6-15% in energy conservation for the short-range communication. This was because the start-up energy was comparable to the transmission energy for short-range communication.

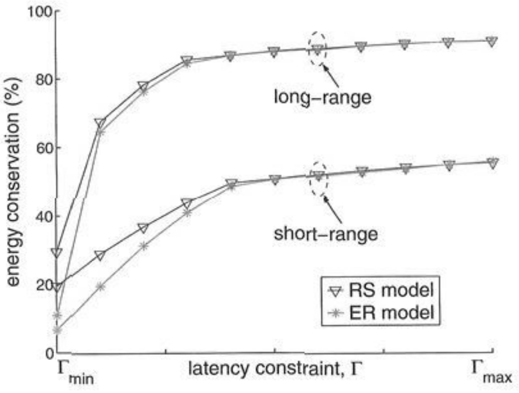


Fig. 5.7 Performance of the on-line protocol (correlation parameter $c = 0.5$, connectivity parameter $\rho = 0.15$, number of sources $N = 30$, sensing range $S = 0.2$).

5.7.3 *Performance of the On-Line Protocol*

We focus on the results for the RS model; similar analysis can be made for the ER model.

5.7.3.1 *Performance Overview*

The energy conservation achieved by the on-line protocol is illustrated in Figure 5.7. The simulated on-line protocol was based on modulation scaling where available modulation levels were even integers within $[2, 8]$. 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 generated using GIT.

When the latency constraint approached Γ_{min}, there was slight performance degradation, compared to DP-Algo from Figure 5.5. Specifically, for the RS model, we observed around 4% less energy conservation for long-range communication, and 3% for short-range communication. This was reasonable, considering that only 4 values were available for the transmission time of each sensor in the on-line protocol, instead of the fine granularity adjustment of the transmission time in DP-Algo. The on-line protocol actually outperformed the modulation scaling case (MS) shown in Figure 5.5, implying a large performance degradation of the rounding technique used by MS.

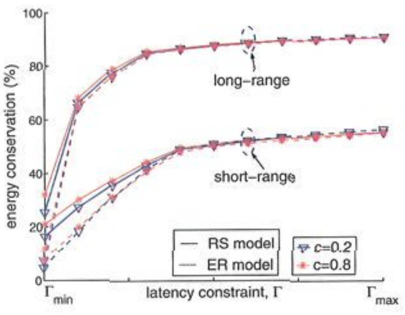
5.7.3.2 *Impact of Network Parameters*

We investigated the impact of network parameters on the performance of the on-line protocol. These parameters included correlation factor c, number of sources N, sensing range S, and the connectivity parameter ρ.

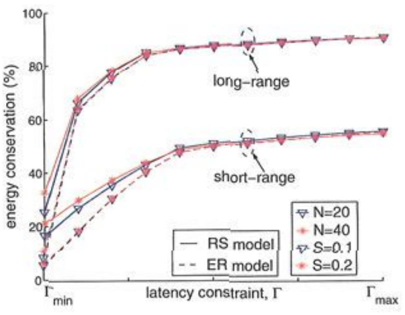
Figure 5.8(a) shows the energy conservation achieved by our online protocol with respect to variations in c and Γ. It was observed that, for a fixed Γ, the resulting energy gain slightly increased with c. This was because a smaller value of c caused larger data packets after aggregation. Thus, the energy cost of links close to the sink node dominated the overall energy cost of the tree. It was, however, difficult to reduce the energy cost of these links, since they had a high likelihood of lying on the longest path of the tree.

Figure 5.8(b) plots the performance of our protocol with respect to variations in N and Γ. It can be seen that when Γ was close to Γ_{min}, the energy gain of the protocol increased with the number of sources. This was because a larger number of sources offers more opportunities for the optimization of links on paths other than the longest one.

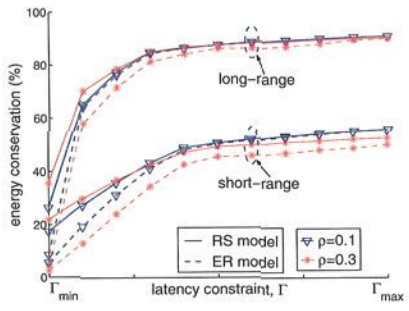
Figure 5.8(c) illustrates the performance of our protocol with respect to variations in ρ and Γ. The energy savings increased with ρ. This was



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



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



(c) Impact of the connectivity parameter ρ ($c = 0.5$, $N = 30$, $S = 0.15$)

Fig. 5.8 Performance of the on-line protocol (c: correlation parameter, ρ: connectivity parameter, N: number of sources, S: sensing range).

understandable, since a large ρ reduced the height of the data gathering tree (the extreme case is a star-like tree formed by setting $\rho = 1$).

5.7.3.3 *Adaptability to System Variations*

Our simulations were performed based on the tree shown in Figures 5.4(a), which had 44 sensor nodes. Thirty of these nodes were chosen as source nodes. Again, we assumed that modulation scaling was used by all the nodes with the available modulation levels being even numbers between 2 and 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, A and B, were investigated to demonstrate the efficiency and adaptability of our protocol.

Scenario A: We fixed s at 200 bits, with $\Gamma_{min} \approx 1.5$ 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 reality, such variations can be caused, for example, by changes in user requests.

We depict the energy cost and latency for data gathering over 4 seconds in Figure 5.9(a), where the optimal solutions were obtained using EMR-Algo. When Γ was fixed, the actual energy cost gradually decreased until it was close to the optimum, while the latency approached the constraint. At 1 second, Γ was varied from 3 milliseconds to 2.1 milliseconds, which caused a violation of the latency constraint. Due to the feedback mechanism, the transmission latency dramatically decreased as the modulation settings of all the sensor nodes were restored to higher levels. Consequently, the energy cost was increased. After that, the energy cost dropped again as time advanced. Moreover, by setting $\beta = 10$, the modulation levels of the sensor nodes were restored to the highest levels when a violation was detected, reflected by the high peaks in the energy curve.

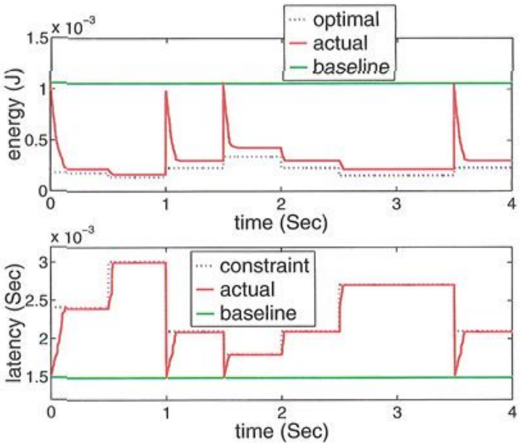
Scenario B: We set $\Gamma = 2.1$ milliseconds, while setting s to 200, 250, 300, 200, 150, 200, and 250 at 0, 0.5, 1, 1.5, 2, 2.5, and 3.5 seconds, respectively. In reality, the change of packet size may be caused by variations in gathered information, or the correlation parameter at sensor nodes. The results are illustrated in Figure 5.9(b), where the optimal solutions were also obtained using EMR-Algo. An analysis similar to that in scenario A can be performed.

In short, our on-line protocol was 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 was also demonstrated.

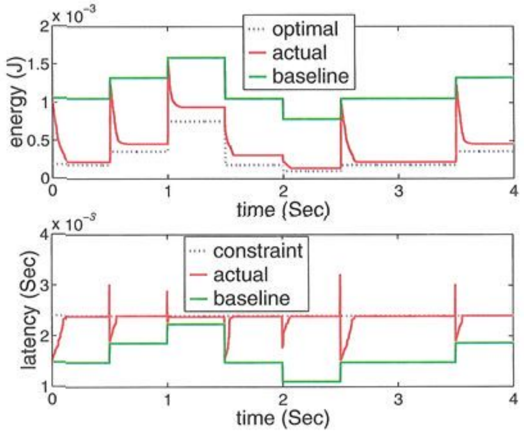
5.8 Summary

In this chapter, we have studied the problem of scheduling packet transmissions over a data aggregation tree, by exploring the tradeoffs of energy *vs* latency using rate adaptation. 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. We have also analyzed the performance of the numerical algorithm for a special case over a complete binary data gathering tree, which matches our simulation results very well. Further, we have proposed a distributed on-line protocol that relies on only local information.

We have presented extensive simulation results for both the off-line and on-line techniques, with respect to variations in several key system parameters. Our simulation results show that between 15% to 90% energy savings can be achieved by the proposed techniques. We have also demonstrated the ability of the protocol to adapt the packet transmission time to variations in the system parameters through two run-time scenarios.



(a) Scenario A



(b) Scenario B

Fig. 5.9 Adaptability of the on-line protocol (correlation parameter $c = 0.5$, connectivity parameter $\rho = 0.15$).

This page is intentionally left blank

Chapter 6

Information Routing with Tunable Compression

6.1 Overview

In the last chapter, we assumed that maximum compression was used for data aggregation. In other words, the data was compressed as much as possible to reduce 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 cases of more advanced and computationally intensive applications with heavy data flow (including streaming media, video surveillance, and image-based tracking), compression of a complex data set is envisioned to have an energy cost comparable to that of wireless communication. A similar situation arises when temporal 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 may not always lead to minimal energy cost. Alternative methods for performing data compression need to be exploited, so that the computation cost can be efficiently traded for the communication cost.

To this end, we use the technique of tunable compression, described in Section 3.2.3, 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 this problem as the Tunable Data Gathering (TDG) problem. As mentioned in Chapter 2, two important data compression schemes have been previously investigated in literature: distributed source coding [172] and compression with explicit communication [47]. The techniques discussed in this chapter are based on the second scheme — to perform joint data compression requires the availability of side information from other sources via explicit communication.

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 the parameters related to the effective sampling size at the sampling stage, and the quantization scaling at the scalar quantization stage, prior efforts have studied the tradeoffs between the quality of the compressed image, and the computation and communication energy costs [182]. Our research complements past work by further exploring the tradeoffs between computation and communication energy at the lossless binary encoding stage.

6.1.1 *Technical Overview*

While most prior work on data gathering focuses on minimizing the communication cost only, our distinguishing goal is to minimize the sum of both computation and communication costs by utilizing tunable compression. To facilitate the tuning of data 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. The TDG problem involves two related subproblems: (1) to construct a data gathering tree and (2) to determine the flow over all paths in the tree. Consider the special case of TDG, where computation energy is negligible, and every node always compresses all of its incoming data to one unit. Since such a case is exactly the Minimal Steiner Tree (MST) problem, TDG is generally NP-Hard.

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 the performance of two existing tree construction methodologies, namely the Shortest Path Tree (SPT), and 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 and data correlation is high. Moreover, MST provides a constant-factor approximation for the grid deployment, regardless of the relative computation cost and data correlation.

We also examine the performance of an approximated MST (referred to as A-MST, due to the NP-Hardness of MST for general graphs) 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 spatial correlation among source data, 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, due to its simplicity, A-MST is suitable as a practical solution. For theoretical completeness, we also present a randomized tree construction methodology that achieves polylogarithmic approximation for general graphs.

6.1.2 *Chapter Organization*

Related work is briefly discussed in Section 6.2. In Section 6.3, we give assumptions and models of our problem, which is formally defined in Section 6.4. We then show how to determine the optimal flow for a given tree in Section 6.5, which enables the performance analysis of SPT and MST on a grid deployment in Section 6.6. In Section 6.7, a randomized approximation algorithm is described. Simulation results are presented in Section 6.8. We summarize this chapter in Section 6.9.

6.2 Related Work

The problem of constructing an energy-efficient data gathering tree in wireless sensor networks, while considering data compression, is gaining increasing research attention. A description of several practical schemes for tree construction is presented in [135]. 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 the maximum possible compression using an MST-like routing among source nodes before routing to the sink; and cluster based routing is a hybrid scheme of RDC and CDR. The performance metric is the accumulated number of bits transmitted over each hop. Assuming that each source node generates one unit of information, the study is motivated by two extreme cases. In the case of zero data correlation, by selecting the shortest path from each source node to the sink node, the optimal solution is the SPT. In the case of perfect data aggregation, exactly one unit of data is routed along each edge of the tree, implying that the optimal solution is the MST. Between these two extreme cases, the optimal tree structure resembles 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 an MST structure within each cluster, and an SPT to route compressed data from each cluster to the sink. A surprising result is that, in a grid-based scenario, while the optimal cluster size depends on the correlation factor, 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 a hybrid routing scheme is confirmed by several other results, under various assumptions to model the data correlation. Cristescu *et al.* assumes a simplified compression model, where the aggregation factor of a piece of information depends only on the availability of side information [47]. Assume that each sensor node in the network generates one unit size of information. Whenever there is 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 an 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 also to the sink, which favors a Minimal Spanning Tree (MSP). Therefore, the optimal solution lies between an SPT and an MSP. In fact, the Shallow Light Tree (STL) proposed by Bharat-Kumar *et al.* [23] can be used to provide a constant factor approximation of the considered problem.

While STL provides an approximation for both SPT and MSP, 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 a tree structure that simultaneously approximates SPT and MST. Consider a more general objective function that minimizes the sum of the 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.* [123] to achieve a $\log k$ performance bound, where k is the number of source nodes.

A simplified version of the algorithm in [123] 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 [67]. The network topology is assumed to be a complete graph, with the 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. It is noted that the concavity assumption for data aggregation function essentially leads to an identical abstraction of the information routing problem with that of a single-source buy-at-bulk problem [13]. 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 with the sink. Another randomized algorithm for information routing on a grid of sensor nodes is proposed by Enachescu *et al.* [59], which is proved to have a constant approximation of the optimal performance.

Very few previous papers have explored the tradeoffs between computation and communication for data gathering [3]. For prediction-based data gathering over a one-dimensional random Gaussian field, such tradeoffs are enabled by adjusting the group size, within which prediction is performed — large groups increase computation cost but decrease communication cost [3]. Simulation results indicate that the optimal group size increases with its distance to the sink. To the best of the authors' knowledge, the technique presented in this chapter is the first work that formally models and studies the tradeoffs between computation and communication energy in a general problem setting. 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 a large volume of data.

Although we still model joint entropy to be a concave function of the number of source nodes, the results in [67] and [13] 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 [18], a randomized algorithm gives an expected $O(\log^2 v)$ approximation of the optimal 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 $\log v \log\log v$ [19] or $\log v$ [62]. However, our major purpose is to illustrate the tradeoffs between SPT and MST, hence the results in [18] suffices.

6.3 Models and Assumptions

6.3.1 *Nomenclature*

A list of notations used in this chapter is given in Table 6.1.

Table 6.1 List of notations.

$NG =< \mathbb{V}, \mathbb{L} >$	The graph representing the underlying network
$\mathbb{R}$	Set of source nodes, $\mathbb{R} \subseteq \mathbb{V}$
w_i	Weight of edge $L_i \in \mathbb{L}$
sink	The sink node in $\mathbb{V}$
δ_i	Number of source nodes in a subtree rooted at $V_i \in \mathbb{V}$
p_i	The path from $V_i \in \mathbb{R}$ 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 normalized by the communication energy cost
f_e^u	Flow from node u on edge e, sometimes simplified to f_e or f
$g(s, f)$	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., $\rho = H_1$, lower bound on the output of compressing one unit data
B_i	When jointly compressed with $i-1$ pieces of unit data, i.e., $B_i = \frac{H_i}{i}$
M, N	Metric spaces on node set $\mathbb{V}$
$d_M(u_1, u_2)$	Distance between nodes u_1 and u_2 on metric space M

6.3.2 *Network Model*

We assume a simplified communication mechanism with a medium access control (MAC) protocol that ensures no packet collisions or interference in the network [176; 148]. Thus, the underling network is modeled as an arbitrary network topology (Section 3.1.2): A graph representation $NG =< \mathbb{V}, \mathbb{L} >$ is used to abstract the underlying network with v sensor nodes and l communication links between nodes. As described in Section 3.1.2, the communication cost over each link is simply abstracted as a scalar valued weight associated with the link, which 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 $sink \in \mathbb{V}$ denote the sink node and $\mathbb{R} \subseteq \mathbb{V}$ denote the set of source nodes.

A data gathering tree is a subtree of NG rooted at *sink* that contains $\mathbb{R}$, denoted as $T =< \mathbb{V}', \mathbb{L}' >$, where $\mathbb{R} \subseteq \mathbb{V}' \subseteq \mathbb{V}$ and $\mathbb{L}' \subseteq \mathbb{L}$. 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 3.1.1, for two nodes $V_1, V_2 \in \mathbb{V}'$, $V_1 \preceq V_2$ indicates that V_1 is a predecessor of V_2. Similarly, for two edges $L_1, L_2 \in \mathbb{L}'$, $L_1 \preceq L_2$ indicates that L_1 is a predecessor of L_2.

6.3.3 *Flow-Based Data Gathering*

Given a data gathering tree over a sensor network, we model data transmission over the tree as a composition of different data flows from each source node to *sink*. That is, each path from a source node to *sink* in the tree corresponds to a data flow over the path. The flow size may change along its corresponding path due to data compression performed by intermediate nodes. The energy cost of the system is the sum of the computation and communication costs of all paths in the tree.

Consider an arbitrary path p_i in the tree, from a source node $V_i \in \mathbb{R}$ to *sink*. Let f_e^i denote the flow over $e \in p_i$ and z_i denote the last edge in p_i, i.e, the edge incident to *sink* in p_i. In the following discussion, since we only consider path p_i, we simplify f_e^i to f_e. We assume that the total energy spent on data compression over the path p_i is determined by the flow on z_i, i.e., the total energy cost for data compression over p_i is calculated as $\frac{\gamma}{f_{z_i}}$. We will justify this assumption in Section 6.3.4.

For a given node in the tree, the number of incoming flows equals the number of source nodes in its subtree. The output size for compressing each incoming packet is lower bounded by the joint entropy of these source nodes. Following the entropy model in [67] (which also effectively abstracts the entropy models in [47; 135]), 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 all nodes 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 maximum 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 $e = (a, b) \in p_i$, we have $f_e \geq B_{\delta_a} = \frac{H_{\delta_a}}{\delta_a}$ (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 $i > 1$. Thus, when a data flow is compressed and transmitted along p_i, the lower bound on the flow decreases as the packet approaches *sink*.

6.3.4 *Discussion*

We discuss the motivation and reasoning for several key assumptions in the TDG problem. These assumptions are used to specify the compression energy model, the data flow model, and the joint entropy model.

First, although the energy function for tunable compression is given by (3.7) in Section 3.2.3, our analysis is not restricted to a specific $g(f)$. In fact, while the energy characteristics of various compression algorithms have been studied in [16], 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 (e.g., Figure 3.4(b)). By using (3.7), the abstraction that the energy cost is inversely proportional to the compression ratio leads to such a convexity. We expect other models to be investigated in this context.

Second, the above flow model naturally models the data streaming from sources to the sink, and facilitates the computation of compression energy cost. This chapter considers only energy cost under this flow model. Other 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.

The presented techniques are 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 sophisticated joint entropy models is the goal of future work. We have assumed that H_i is 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 6.5.

Third, the assumption of determining the compression energy over a path based solely on the flow on the last edge in the path ignores the cost 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 [16], techniques such as *gzip* consume 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. To model the computation cost more accurately is part of our future research.

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 edge weights.

6.3.5 *An Example*

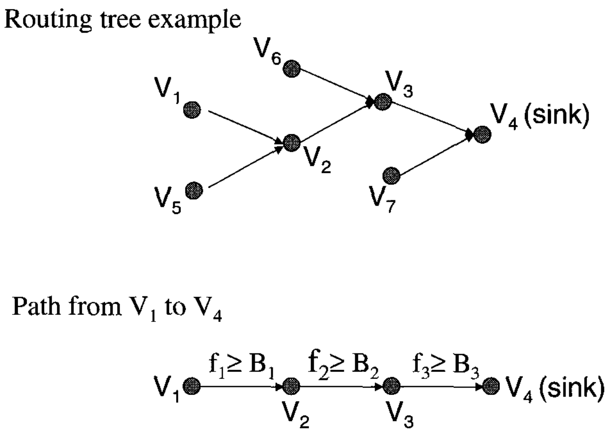


Fig. 6.1 An example data gathering tree and a path within it.

To illustrate the flow model in Section 6.3.3, consider the data gathering tree given in Figure 6.1, 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*. We have a total of 4 paths in this tree.

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 (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. 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 the lower bounds of flow on each link, are also illustrated in Figure 6.1 (the superscript for f_i^1 is omitted in the figure). Based on our model, we also have $B_1 \geq B_2 \geq B_3$.

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). 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). The flow on the link from V_7 to *sink* is lower bounded by B_1.

6.4 Problem Definition

Based on the models and notations in Section 6.3, our *Tunable Data Gathering (TDG)* problem is formally defined:

Given:

(i) a weighted graph $NG =< \mathbb{V}, \mathbb{L} >$ *with weight* w_i *for each* $L_i \in \mathbb{L}$, *sink* $\in \mathbb{V}$, $\mathbb{R} \subseteq \mathbb{V}$,

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

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

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

$$\sum_{V_i \in R} \left(\frac{\gamma}{f^i_{z(i)}} + \sum_{e \in p_i} f_e^i w_e\right) \tag{6.1}$$

subject to

$$\forall V_i \in R, \forall e = (a,b) \in p_i \Rightarrow f_e^i \geq B_{\delta_a} \text{ , and} \tag{6.2}$$

$$\forall v \in R, \forall e_1 \prec e_2 \in p(v) \Rightarrow f_{e_1}^v \geq f_{e_2}^v \text{ ,} \tag{6.3}$$

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

We consider two special cases of the TDG problem. In the first case, we assume $\gamma = \infty$ or $H_i = i$, i.e., when computation energy is arbitrarily high or data is uncorrelated. Either condition leads to the obvious solution that no compression is performed in the data gathering tree. We construct the SPT of G 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 $H_i = 1$. In other words, computation energy is negligible and the joint entropy of any arbitrary i source nodes is always one. The desired flow on all edges of the tree is unity. Apparently, MST gives the optimal tree construction. Obviously, certain tradeoffs exist between SPT and MST.

Since the second special case requires the construction of an MST, the TDG problem is generally NP-Hard. To cope with this NP-Hardness, we begin our study by decoupling the two subproblems of selecting the tree construction and determining the flow from each source node to the sink. In the next section, we show the optimal tunable compression strategy for a path in a given tree.

6.5 Optimal Flow in a Given Tree

Given a data gathering tree and an arbitrary source node $u \in \mathbb{R}$, 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 $V_1 = u$, $V_k = sink$, and k is the number of nodes along the path. We need to compress and transmit a packet of unit size from V_1 to V_k using the minimal computation and communication energy costs. 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 superscript of elements in vector $\vec{f}$, as well as e in the subscript. 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 6.3.5). 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 6.3.3, we have $\beta_i \leq \beta_{i-1}$.

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 $W_k = 0$.

We first gain insight into the optimal flow on the path by revisiting the example in Section 6.3.5. Then, we will formally prove the optimal flow.

6.5.1 *Example Revisited*

We consider the flow on path V_1 to V_4 in Figure 6.1. Denote the flow as $\vec{f} = \{f_1, f_2, f_3\}$. For this flow, we have $\beta_1 = B_1$, $\beta_2 = B_2$, and $\beta_3 = B_3$.

Intuitively, as the relative computation cost increases, the optimal solution performs less compression. In the trivial case when the computation cost is prohibitively high, i.e., $\gamma \geq W_1$, 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 and w_i:

(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 $f_3 \in [\beta_3, \beta_2]$.

The optimal flow behaves as a piece-wise function of γ and w_i, which abstracts the relative cost of computation. The optimal flow for the above example is summarized in Table 6.2.

Table 6.2 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	$\frac{\gamma}{\beta_2} < \beta_2 W_3$	$f_1 = \beta_1$, $f_2 = \beta_2$, $f_3 \in [\beta_3, \beta_2]$

6.5.2 *Determining the Optimal Flow*

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

Lemma 6.1 *For any optimal flow $\vec{f}$ over the path $p = \{V_1, V_2, \ldots, V_k\}$, 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, contradicting the optimality of the flow. □

Theorem 6.1 *Given a path $p = \{V_1, V_2, \ldots, V_k\}$, 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 $i \in [k-1]$. Then, the optimal flow $\vec{f}$ is:*

$$\vec{f}(\gamma) = \{\beta_1, \beta_2, \ldots, \beta_{i-2}, \beta_{i-1}, \underbrace{f^*, \ldots, f^*}_{k-i}\}, \tag{6.4}$$

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

Proof.

If $\gamma \geq W_1$, then any compression is more expensive than transmitting the original data along the path. Thus, the optimal solution is simply to 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 $[0, W_1]$.

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 $\max\{\beta_i, \sqrt{\frac{\gamma}{W_i}}\}$. 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

$$\begin{aligned}
\epsilon_x - \epsilon_f &= \left(\frac{\gamma}{x_{k-1}} + \sum_{j=1}^{k-1} x_j w_j\right) - \left(\frac{\gamma}{f^*} + \sum_{j=1}^{i-1} x_j w_j + f^* \sum_{j=i}^{k-1} w_j\right) \\
&= \frac{\gamma}{x_{k-1}} + \sum_{j=i}^{k-1} x_j w_j - \left(\frac{\gamma}{f^*} + f^* W_i\right) \\
&\geq \left(\frac{\gamma}{x_{k-1}} + x_{k-1} W_i\right) - \left(\frac{\gamma}{f^*} + f^* W_i\right).
\end{aligned} \tag{6.5}$$

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

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

Equation (6.5) is actually $P(x_{k-1}) - P(f^*)$. 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 $\sqrt{\frac{\gamma}{W_i}} \geq \beta_i$, implying that $f^* = \sqrt{\frac{\gamma}{W_i}}$. For the above optimization problem $P(y)$, we have $P'(\beta_i) = -\frac{\gamma}{\beta_i^2} + W_i \leq 0$ and $P'(\beta_{i-1}) = -\frac{\gamma}{\beta_{i-1}^2} + W_i \geq 0$, where $P'(y)$ is the first derivative of $P(y)$. Therefore, the optimal y that leads to $P'(y) = 0$ lies within $[\beta_i, \beta_{i-1}]$. By solving $P'(y) = 0$, we know that the optimal value of y actually equals f^*. Thus, $\epsilon_x - \epsilon_f = P(x_{k-1}) - P(f^*) \geq 0$, implying that $\vec{f}$ is optimal. From Lemma 6.1, and the fact that $f^* \in [\beta_i, \beta_{i-1}]$, 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 $P'(\beta_i) = -\frac{\gamma}{\beta_i^2} + W_i \geq 0$. This means that $P(y)$ is an increasing function when $y \geq \beta_i$. Thus, the value of y that minimizes $P(y)$ is 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 6.1, the optimal flow is trivial when $\gamma \geq W_1$. Therefore, we focus on the case when $\gamma < W_1$ in the following discussion.

Theorem 6.1 reveals the fact that, for an optimal flow from V_1 to *sink*, if $\gamma \in [W_{i+1}\beta_i^2, W_i\beta_{i-1}^2]$ for some $i \in [k-1]$, the flow on the last $k - i$ edges is equal to f^*. For a closer understanding of f^*, in Figure 6.2, we plot f^* as a function of γ for the example path from V_1 to V_4 (Figure 6.1), by setting $w_1 = w_2 = w_3 = 1$, $\beta_1 = 0.7$, $\beta_2 = 0.6$, and $\beta_3 = 0.5$. The labels on the x-axis are $a_1 = W_3\beta_3^2$, $a_2 = W_3\beta_2^2$, $a_3 = W_2\beta_2^2$, $a_4 = W_2\beta_1^2$, and $a_5 = W_1\beta_1^2$. It can be observed, for example, that when $\gamma \in [a_5, W_1]$, f^* equals $\sqrt{\frac{\gamma}{W_1}}$. When γ is decreased to within $[a_4, a_5]$, f^* is, however, lower bounded by β_1, as indicated by Theorem 6.1.

Let $Diam(sink, \mathbb{R})$ denote the weighted diameter of G with respect to R and *sink*, i.e., the maximum among the shortest weighted path from any node in R to *sink*.

Corollary 6.1 *Given G, if $\gamma \geq Diam(sink, \mathbb{R}) \times B_1^2$, the Shortest Path Tree is the optimal tree for the TDG problem, with the flow specified by Theorem 6.1.*

Proof. From Theorem 6.1, the optimal flow from any $u \in \mathbb{R}$ along its shortest path to *sink* equals some fixed value within $[1, B_1]$ for all edges along the path. Thus, joint data compression from u with other source

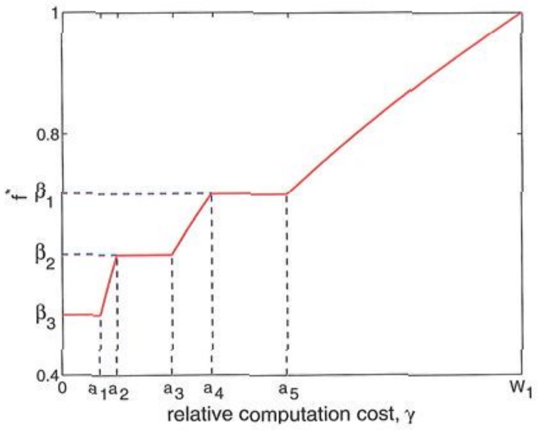


Fig. 6.2 f^* for the example path in Figure 6.1 as a function of γ.

information only decreases the lower bound of the feasible flow, but never reduces the cost. □

Let $\gamma^* = Diam(sink, \mathbb{R})B_1^2$. We refer to γ^* as the *critical point* of the problem.

6.6 Analytical Study of SPT and MST

6.6.1 *Analysis for a Grid Deployment*

For analytical tractability, we assume a grid deployment of size $r \times 2r$, where r source nodes at the leftmost column need to send information to the *sink*, located at the bottom right corner of the grid (similar network deployments have also been studied in [135; 156] for tractable analysis). Each sensor node can communicate with its one hop neighbors, i.e., 8 neighbors when ignoring boundary effects. We also assume $w_i = 1$ for all $e_i \in E$.

The routing constructed by SPT and MST are illustrated in Figure 6.3. Note that although to find an MST for a general graph is NP-Hard, the MST for the specific grid deployment in Figure 6.3 is trivial. From Corollary 6.1, the SPT is optimal when $\gamma \geq \gamma^* = (2r-1)B_1^2$. We are therefore only interested in the performance of the SPT and MST for $\gamma \in [0, \gamma^*]$.

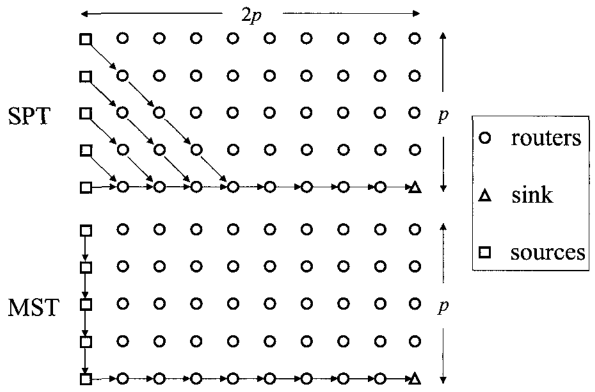


Fig. 6.3 SPT and MST routing schemes for a grid.

Let ε_{SPT} denote the energy cost for SPT and ε_{MST} for MST. Using Theorem 6.1, ε_{SPT} can be calculated as:

$$\varepsilon_{SPT} = \begin{cases} \frac{r(r-1)B_1}{2} + i(\frac{\gamma}{f^*} + f^*(2r-i)) + \sum_{j=1}^{i-1} jB_j \\ \quad + \sum_{j=r}^{2r-i-1} (\frac{\gamma}{f'} + jf'), \\ \quad \text{when } \gamma \in [(2r-i-1)B_i^2, (2r-i)B_{i-1}^2] \\ \quad \text{for some } 1 \le i \le r & \text{(6.6a)} \\ \frac{r(r-1)B_1}{2} + \frac{r\gamma}{B_r} + r^2 B_r + \sum_{j=1}^{r-1} jB_j, \\ \quad \text{when } \gamma \in [0, (r-1)B_r^2] & \text{(6.6b)} \end{cases}$$

where $f^* = \max\{B_i, \frac{\gamma}{2r-i}\}$, and $f' = \min\{B_1, \frac{\gamma}{j}\}$. Note that for (6.6a), the upper bound of γ equals $(2r-1)$ when $i = 1$, which is slightly larger than γ^*. However, this does not affect our further analysis.

Intuitively, (6.6a) can be understood by the example shown in Figure 6.4. The cost $\frac{r(r-1)B_1}{2}$ is for packet transmission over edges in triangle A_1. The term $i\frac{\gamma}{f^*}$ corresponds to the computation cost of the i source nodes circled in area A_2, which have an optimal flow ended with f^* on their paths to the sink. The cost $i(2r-i)f^*$ is for transmitting flow f^* from the i source nodes in A_2 over their last $2r-i$ hops in A_4. The cost $\sum_{j=1}^{i-1} jB_j$ is for

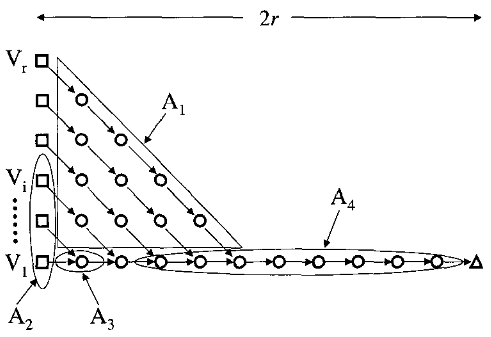


Fig. 6.4 Decomposition of ε_{SPT}.

packet transmission from the i nodes in A_2 over edges in A_3. The term $\sum_{j=r}^{2r-i-1}(\frac{\gamma}{f'}+jf')$ is the compression cost for the $r-i$ source nodes not in A_2, plus the cost of packet transmission from these source nodes over edges in A_4. The min function in f' is due to constraint (6.3).

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

$$\varepsilon_{MST} = \begin{cases} 2\sqrt{\gamma}\sum_{j=2r-1}^{q-i-1}\sqrt{j}+\sum_{j=1}^{i-1} jB_j \\ \quad +i(\frac{\gamma}{f^*}+f^*(q-i)), \\ \quad \text{when } \gamma \in [(q-i-1)B_i^2,(q-i)B_{i-1}^2] \\ \quad \text{for some } i^* \leq i \leq r & (6.7a) \\ \frac{r\gamma}{B_r}+\sum_{j=1}^{r-1} jB_j + r(2r-1)B_r, \\ \quad \text{when } \gamma \in [0, 2rB_r^2] & (6.7b) \end{cases}$$

where $f^* = \max\{B_i, \sqrt{\frac{\gamma}{q-i}}\}$.

Also, the minimal cost of the TDG problem is lower bounded by replacing constraint (6.2) with $\forall v \in R, \forall e \in p(v), f_e^v \leq B_{|\mathbb{R}|}$, with $|\mathbb{R}| = r$ in this particular case. In other words, we assume that distributed source coding among all source nodes is available at no extra cost. It can be verified that the optimal routing for such a lower bound case forms exactly an SPT. Thus, the energy costs for the lower bound, ε_{LB} can be calculated as

$$\varepsilon_{LB} = \begin{cases} 2r\sqrt{\gamma(2r-1)}, & \\ \qquad \text{for } (2r-1)B_r^2 \le \gamma \le \gamma^* & (6.8a) \\ \frac{r\gamma}{B_r} + r(2r-1)B_r, & \\ \qquad \text{for } 0 \le \gamma \le (2r-1)B_r^2 & (6.8b) \end{cases}$$

Using the example of ε_{SPT}, the explanation of ε_{MST} and ε_{LB} is simple, and omitted here.

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

Observation 1 For the grid deployment in Figure 6.3, we have the following performance bound regarding SPT and MST (refer to Appendix B for proof):

$$\lim_{\gamma \to \gamma^*} \frac{\varepsilon_{SPT}}{\varepsilon_{LB}} = O(1) \tag{6.9}$$

$$\lim_{\gamma \to 0} \frac{\varepsilon_{SPT}}{\varepsilon_{LB}} = O(\frac{r}{H_r}) \tag{6.10}$$

$$\lim_{\gamma \to \gamma^*} \frac{\varepsilon_{MST}}{\varepsilon_{LB}} = O(1) \tag{6.11}$$

$$\lim_{\gamma \to 0} \frac{\varepsilon_{MST}}{\varepsilon_{LB}} = O(1) \; . \tag{6.12}$$

where $\gamma^* = (2r-1)B_1^2$ is the critical point of the system.

The main lesson from this observation is that, for this particular grid deployment, the energy cost of MST is a constant approximation of the optimal cost, regardless of the form of H_i and the relative energy cost γ. Although theoretically the performance of MST for general graphs is unbounded in the worst case, it is natural to conjecture that MST might also perform well on average. In fact, our simulation results suggest that an approximated MST can be used as a practical routing scheme for solving the TDG problem with acceptable performance on average.

We also notice that when γ approaches 0, the ratio of ε_{SPT} to ε_{LB} is $O(\frac{r}{H_r})$, which indicates that the performance of SPT improves when correlation among sources decreases. In the special case when $H_r = \Theta(r)$, SPT becomes the optimal structure.

We verify the above observation through numerical results in the next section.

6.6.2 *Tradeoffs Between SPT and MST*

For the purpose of demonstration, we instantiate H_i using a set of practical joint entropy models from [117]. Specifically, we consider a stationary Gaussian random process using a scalar quantizer, with uniform step size and an infinite number of levels. Our entropy models fall into 3 classes, where d is the distance between source nodes:

E1: When the correlation coefficient is e^{-d^2}, H_i scales as $O(\log i)$ as $i \to \infty$ [117].

E2: When the correlation coefficient is e^{-d}, H_i scales as $O(\sqrt{i}\log i)$ as $i \to \infty$ [117].

E3: When all sources are independent of each other, H_i scales as $O(i)$ as $i \to \infty$.

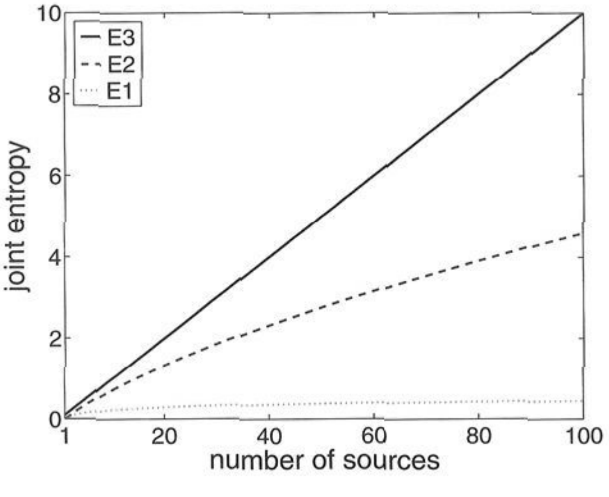


Fig. 6.5 Entropy models E1, E2, and E3.

We set $H_1 = \rho$, where ρ is the data entropy rate. According to the lossless compression ratio for CCITT test images [41], we set $\rho = 0.1$. For $i > 1$, we set H_i to $\rho \log i$ for E1, $\rho\sqrt{i}\log i$ for E2, and $i\rho$ for E3. [1] The three entropy functions are depicted in Figure 6.5. Intuitively, the correlation among sources is highest in the case of E1, and lowest in the case of E3.

[1]While H_i is asymptotically bounded by these functions, we set B_i to B_{i+1} if $B_i < B_{i+1}$ for small i's in our numerical calculation to sustain our assumption that $B_i \geq B_{i+1}$.

We shall see that this difference does affect the tradeoffs between SPT and MST according to Observation 1.

6.6.2.1 *Tradeoffs for Entropy Model E1*

In Figure 6.6, we plot ε_{SPT}, ε_{MST}, and ε_{LB} for E1, with $r = 40$ and γ varied between 0 and 0.4 (note that in this case $\gamma^* = (2r-1)\rho^2 \approx 0.8$).

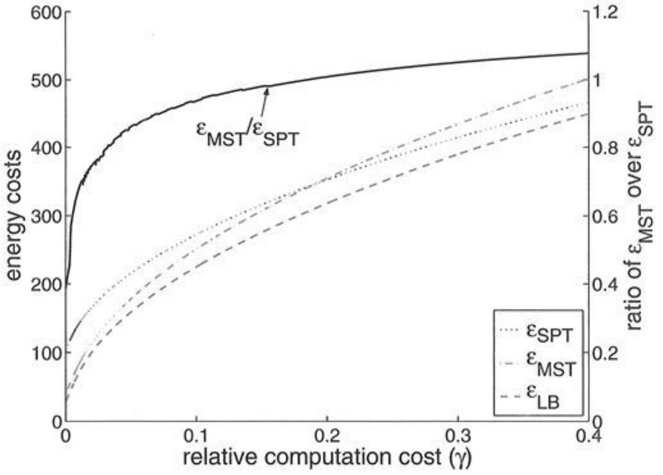


Fig. 6.6 Performance of SPT and MST for grid deployment with entropy model E1.

From Figure 6.6, we observe the tradeoffs between SPT and MST with respect to variations in γ. When γ is large, SPT outperforms MST with ε_{SPT} approaching 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 10% increase over SPT (from our results, ε_{MST} is also within 15% off ε_{SPT} when $\gamma = 0.8$).

On the other hand, when γ approaches to zero, MST provides up to 60% energy reduction compared to SPT. This is because when the computation costs are low, compressing data from multiple sources before routing to the sink provides higher gains by reducing the flow on the tree. In the special case of $\gamma = 0$, our problem becomes similar to the scenario studied in [135], where tradeoffs between MST and SPT exist due to variations in spatial correlation among source nodes — MST outperforms SPT when the correlation is high and SPT outperforms MST when the correlation is low. In the case of E1, the spatial correlation captured by $H_i = O(\log i)$ determines that MST outperforms SPT when $\gamma = 0$, which keeps with the results in [135].

We also examined different scenarios by varying r from 5 to 150; very similar tradeoffs between SPT and MST were observed.

6.6.2.2 *Tradeoffs for Entropy Model E2*

The tradeoffs between SPT and MST are more complicated in the case of E2. Based on Observation 1, the asymptotic ratio $\frac{\varepsilon_{SPT}}{\varepsilon_{LB}}$, as $\gamma \to 0$, approaches $\frac{r}{H_r}$, which is $O(\frac{\sqrt{r}}{\log r})$ in the case of E2. Compared with the $O(\frac{r}{\log r})$ ratio in case of E1, there is an improvement by a factor of $O(\sqrt{r})$, as shown by the following numerical results. Moreover, the ratio $\frac{r}{H_r}$ increases with r, which is also verified by our results.

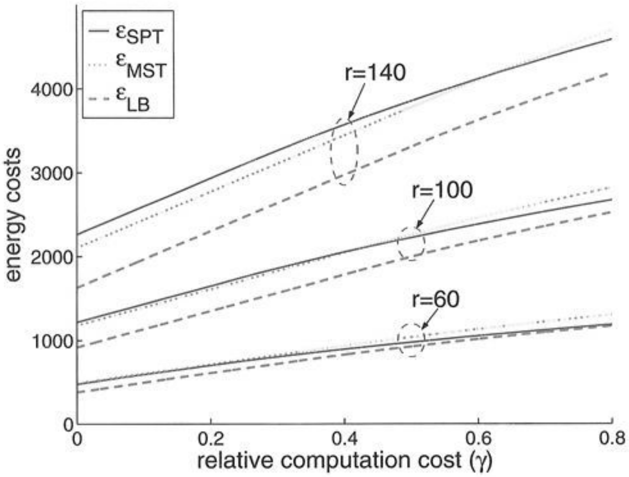


Fig. 6.7 Performance of SPT and MST for grid deployment with entropy model E2.

In Figure 6.7, we illustrate the performance of SPT and MST with respect to variations in both γ and r. In this figure, we vary r from 60 to 140 in increments of 40, and γ from 0 to 0.8. When $r = 60$, both the performance of SPT and MST is very close to ε_{LB}. However, as r increases, we can see the degradation of the performance of SPT. When $r = 140$, the tradeoffs between SPT and MST are apparent, and similar to those in Figure 6.6.

6.6.2.3 *SPT is optimal for Entropy Model E3*

In the case of E3, there is no correlation between any sources. Thus, SPT is the optimal solution. This conclusion is true for not only the analyzed grid deployment, but general graphs as well.

6.6.3 ***Summary of Grid Deployment***

Based on the analysis and numerical results for a grid deployment, we draw the following insights about the choice of SPT or MST as a routing substrate, under different variations of correlation among source information and the relative cost of computation.

(1) When the relative cost of computation is expensive, SPT is preferred.
(2) When the correlation among source information is high and the relative cost of computation is cheap, MST is favored.
(3) When the correlation among source information is low and the relative cost of computation is cheap, the choice depends on the exact characteristics of correlation and number of sources.

In spite of these differences between SPT and MST under various circumstances, Observation 1 indicates that MST delivers constantly bounded performance compared to the optimum, regardless of the variations in γ and H_i. Although the above analysis and numerical results are based on a specific grid topology, they imply reasonably good performance of MST for general graphs, which is demonstrated in Section 6.8.

Another important insight is that when γ varies, the optimal routing scheme to TDG shall explore the tradeoffs between SPT and MST. Along this direction, we exploit in next section a hierarchically clustered tree structure for solving the TDG problem.

6.7 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 [18]. 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 of at most $O(\log^2 v)$ times the optimum.

We first give the notion of probabilistic metric approximations from [18]. Given $\mathbb{V}$ denoting a set of v points, let M denote a metric space over $\mathbb{V}$ where the distance between $u_1, u_2 \in \mathbb{V}$ is denoted by $d_M(u_1, u_2)$.

Definition 6.1 A metric space N over $\mathbb{V}$ dominates a metric space M over $\mathbb{V}$, if for every $u_1, u_2 \in \mathbb{V}$, $d_N(u_1, u_2) \geq d_M(u_1, u_2)$.

Definition 6.2 A metric space N over $\mathbb{V}$ α-approximates a metric space M over $\mathbb{V}$, if it dominates M, and for every $u_1, u_2 \in \mathbb{V}$, $d_N(u_1, u_2) \leq \alpha \cdot d_M(u_1, u_2)$.

Definition 6.3 A set of metric spaces $\mathcal{S}$ over $\mathbb{V}$ α-probabilistically-approximates a metric space M over $\mathbb{V}$, 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 $u_1, u_2 \in \mathbb{V}$, $E(d_N(u_1, u_2)) \leq \alpha \cdot d_M(u_1, u_2)$.

Definition 6.4 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 [18] can be simply stated as follows:

Theorem 6.2 *Every weighted connected graph NG with v vertices 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, until each sub-cluster contains only one node from NG. Details of the construction of the set of k-HST's can be found in [18].

Using these metric approximations, we can claim the following theorem.

Theorem 6.3 *Given a TDG problem on graph NG with optimal cost C, there is a feasible solution on the set of k-HST's that α-probabilistically-approximates 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 T^*. Without loss of generality, let let $p = \{V_1, V_2, \dots, V_k\}$ denote the path from V_1 to *sink*, where V_1 is the source node, k is the number of nodes on the path, and V_k is the *sink*. We use the notations in Theorem 6.1. Let f_j denote the flow on link $e_j = (V_j, V_{j+1})$, where $j \in [k-1]$. From Theorem 6.1, 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 $p_M = \{M_{e_1}, M_{e_2}, \dots, M_{e_{k-1}}\}$. 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, or 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 of 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 $\beta'_k \le \beta_j$. 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:

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

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. □

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. By mapping each route in T to a path in NG, we can construct 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 we need to map every internal node in T to a node 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 in path length resulting from the above mapping is bounded by a factor of k.

We therefore have the following algorithm:

(1) Choose at random a tree $T \in \mathcal{S}$.
(2) Map the route from each $V_i \in \mathbb{R}$ to *sink* on T to a path on NG (which is also from V_i to *sink*) as described in Theorem 6.3.
(3) Determine the optimal flow on each path, based on Theorem 6.1.

Theorem 6.4 *Given a TDG problem on a graph G, the above randomized algorithm gives a* $O(\log^2 v)$ *approximation.*

6.8 Simulation Results

6.8.1 *Simulation Setup*

A sensor network was generated by randomly scattering v sensors in a unit square with a uniform distribution. The communication range of the radios was set to r, which indicates that the average number of neighbors for each sensor node was $v\pi r^2$ (ignoring the boundary effect). The weight on each edge could be modeled by using various path-loss models from wireless communications. In our simulation, we set the weight on each edge to be the geometric distance between the two nodes incident to the edge. The sink node was fixed at the left-bottom corner of the square, while the source nodes were randomly selected from all the nodes in the square. Since SPT is always the optimal solution in E3, we present results for both E1 and E2 entropy models with an emphasis on E1.

The performance of the 3 tree construction methods, SPT, MST, and k-HST (or simply HST hereafter), was studied in our simulations. While SPT and 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 [100], which gave a 2-approximation MST [180], with A-MST denoting the resulting approximated MST. The lower bound of the TDG problem was obtained using the relaxation method described in Section 6.6.

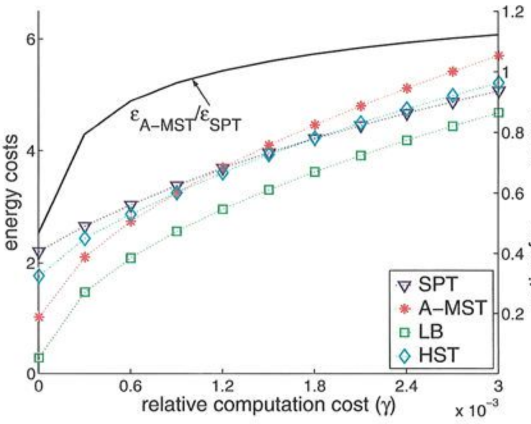
All the data shown in this section was averaged over more than 300 instances, such that they have a 95% confidence interval with a $\leq$ 2% precision. For each instance, the sensor field was randomly generated using the above procedure.

6.8.2 *Results*

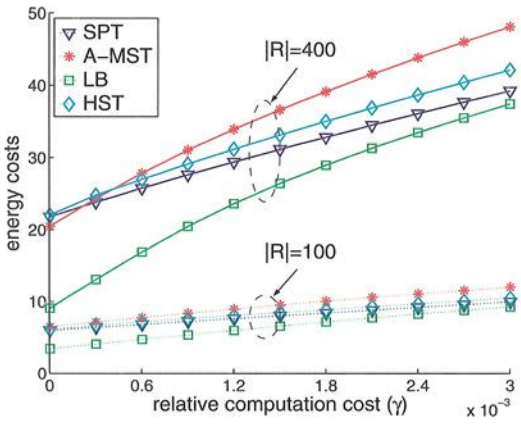
6.8.2.1 *Main Results*

For the results shown in Figure 6.8, we fixed $v = 600$ and $r = 0.2$, while varying γ within $[0, 0.003]$ so that we could focus on the tradeoffs between A-MST and SPT. We observed that the simulation results for general graphs confirmed the analytical tradeoffs between SPT and A-MST in Section 6.6. In the case of entropy model E1, the performance of SPT approached the lower bound as γ increased, while A-MST outperformed SPT when γ tended to zero. As expected, HST performed in between SPT and A-MST throughout the variations in γ. More importantly, A-MST demonstrated acceptable performance throughout the variations in γ. The curve of $\varepsilon_{MST}/\varepsilon_{SPT}$ clearly showed that A-MST offered 50% energy savings over SPT when $\gamma = 0$, and $\leq$ 15% increase over SPT at high γ.

In the case of entropy model E2, we observed the expected performance improvement of SPT over the case of E1. For the sub-case of 200 source nodes, the performances of SPT and A-MST were very close to the lower bound. When the number of sources was increased to 400, the tradeoffs between SPT and A-MST became observable. This conformed with the analytical results for the grid deployment in Figure 6.7, which indicate that the choice between A-MST and SPT depends on the exact entropy model, as well as the number of sources.



(a) Entropy model E1 with 50 source nodes



(b) Entropy model E2 with 200 and 400 source nodes

Fig. 6.8 Main simulation results ($v = 600$, $r = 0.2$).

We also conducted simulations using other values of v, with similar performance trends observed.

The simulation results presented here indicate that when the entropy model and relative computation cost γ are known, either SPT or A-MST can be selected accordingly as a practical routing scheme. When γ is unknown or demonstrates high variations, HST can be used to provide an approximation with a theoretically guaranteed performance bound. Nevertheless, in practice, the simple A-MST performs well on average, with only slight degradation compared to SPT when γ is large, or the correlation among sources is low.

In the following, we focus on the results for entropy model E1.

6.8.2.2 *Impact of the number of source nodes* $|\mathbb{R}|$

For the results shown in Figure 6.9, we fixed $v = 600$, $r = 0.2$, while setting $|\mathbb{R}|$ to 25 or 100 and varying γ within $[0, 0.003]$. The energy costs of all tree structures increased with $|\mathbb{R}|$. Nevertheless, the tradeoffs between SPT, A-MST, and HST still held for different values of $|\mathbb{R}|$.

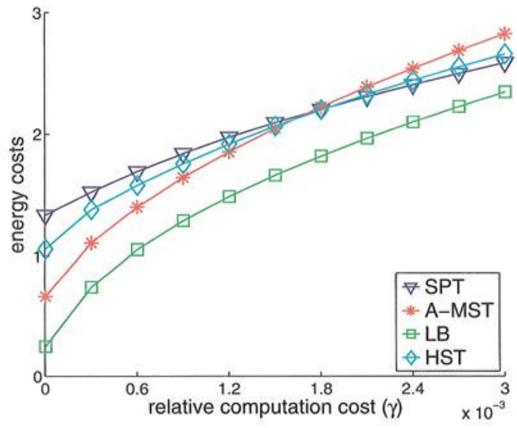
6.8.2.3 *Impact of the communication range* r

For the results shown in Figure 6.10, we fixed $v = 600$, $|\mathbb{R}| = 50$, while setting r to 0.1 or 0.3 and varying γ within $[0, 0.003]$. The tradeoffs between SPT, A-MST, and HST were still apparent for different r. We also observed that increasing r led to a very similar impact on the performance of different tree structures as when v increased. This is because increasing r also results in an increased number of neighbors, which in turn offers opportunities for better tree construction. However, since SPT always routes through the shortest path, which may hinder data aggregation, ε_{SPT} increased with r when $\gamma = 0$.

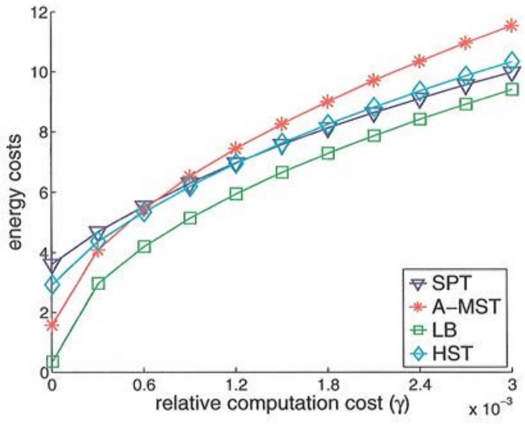
6.9 Summary

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 costs. In the TDG problem, tunable compression is applied to realize the fundamental tradeoffs between computation and communication energy.

We have developed a suitable energy model for tunable compression, and a flow based model for gathering information from correlated sources.



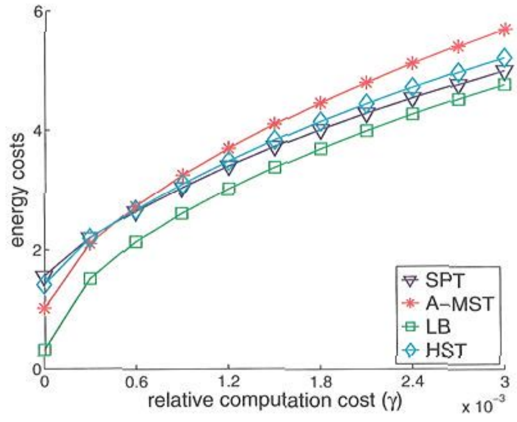
(a) $|\mathbb{R}| = 25$



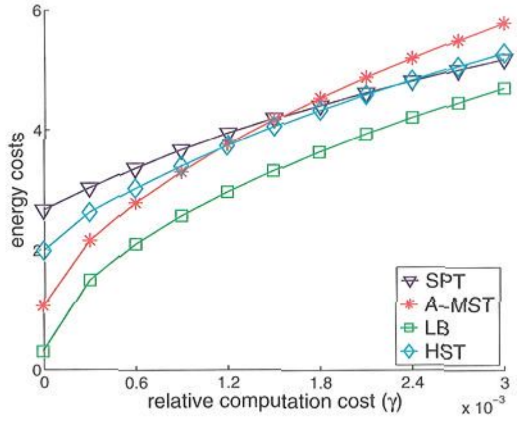
(b) $|\mathbb{R}| = 100$

Fig. 6.9 Impact of the number of source nodes, $|\mathbb{R}|$, and the relative computation cost, γ, on energy costs ($v = 600$, $r = 0.2$).

We have derived the optimal solution for scheduling tunable compression at different nodes on a given data gathering tree. Based on this, we have both analytically and empirically illustrated the tradeoffs between two data gathering trees, namely the Shortest Path Tree (SPT) and the Minimal Steiner



(a) $r = 0.1$



(b) $r = 0.3$

Fig. 6.10 Impact of the communication range, r, on energy costs ($v = 600$, $|\mathbb{R}| = 50$).

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 acceptable performance on average for general graphs. Due to its simple implementation, the A-MST is attractive for practical applications.

One important piece of future work is to identify suitable techniques or algorithms that can be used to realize tunable compression for data gathering, such as for algorithms intended for multimedia processing. It is crucial to develop accurate energy models for these techniques to apply the algorithms described in this chapter. We are interested in exploiting resource-intensive advanced sensor network applications that may actually gain benefits from our work.

This page is intentionally left blank

Chapter 7

Conclusions

7.1 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. These three topics include data processing within a collocated cluster, data transportation over a data gathering tree, and information routing with tunable compression. We summarize the three topics along four dimensions: their functionalities, objectives, performance constraints, and the involved tradeoffs.

From the perspective of application level functionality, the 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 [72; 167; 207; 208]. In such infrastructures, information sensing and processing is performed in a localized fashion in each cluster, while the outputs from all clusters are routed to the base station. By addressing these various stages of information processing and routing, we believe our work provides a framework over which various extensions can be further explored. We are also aware of other schemes for information processing and routing, such as the network flow-based scheme [78; 80; 92]. To apply the presented optimization techniques in the context of such schemes is a very interesting topic.

From the perspective of optimization objectives, we aim to (1) balance the energy cost of sensor nodes for the topic of in-cluster processing, and (2) minimize the overall energy costs for the other two topics — information transportation and routing. While both objectives can be used to improve the energy-efficiency of the system, their advantages and disadvantages still have not been well quantified by any research efforts at this time. In general, balancing the energy cost can provide fairness in the lifetime of sensor nodes. By avoiding the so-called "hot spots" in the network (heavily loaded sensor nodes that die earlier than other nodes due to depleted energy), we gain advantages in terms of network connectivity and coverage. However, balancing the energy cost may not lead to a minimal overall energy cost, and therefore a reduced system lifetime. It has been suggested that in a highly dense network, energy-balance can be realized by other system knobs, such as sleep scheduling. Therefore, we can simply minimize the total energy cost of waking sensor nodes, while still achieving a certain degree of energy balance across the system through sleep scheduling. Future work is needed to justify the tradeoffs between balancing the energy cost and minimizing the overall energy costs.

From the perspective of performance constraints, we have considered real-time latency constraints for both in-cluster processing and information transportation over a given tree substrate. Optimization under such constraints is important for many mission-critical applications envisioned in the near future [22; 25; 115]. 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, and apply them separately. In the topic of information transportation over a given tree, the impact of joint information entropy is addressed by assuming a given aggregation function. The constraint of joint information entropy is explicitly considered while constructing a routing tree with tunable compression. The latency constraint, however, 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, which can be considered in the future.

From the perspective of the involved system knobs, we have presented a unified scheme for the voltage scaling and rate adaptation of in-cluster processing to explore the tradeoff of energy *vs* latency. A similar tradeoff is explored for information transportation over a given tree. Although not explicitly emphasized, the work on in-cluster processing also achieves a certain balance between the computation and communication energy costs. Such a balance is explicitly addressed using tunable compression in 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.

7.2 Future Work

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

7.2.1 *Adaptive Fidelity Algorithms*

Adaptive fidelity algorithms were briefly mentioned in Chapter 6. According to the definition by Estrin *et al.* [60], "an 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." At a higher level of abstraction, the fidelity of the information delivered to end users is one of the application-level performance metrics that can be traded with others, including computation and communication capabilities and energy cost. By considering the adaptive fidelity algorithm, we enlarge the design space of cross-layer optimization by one more dimension.

One example given by Estrin *et al.* [60] is selectively switching off a certain portion of sensor nodes during object localization. While the precision of the object location may be compromised due to fewer sensor nodes being 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 image quality with both processing delay and compression output size [182]. 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 in 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 energy costs. This identification is application-specific, as illustrated by the following two examples.

The first example is the so-called energy scalable algorithm proposed by Sinha *et al.* [171]. An algorithmic transform is performed such that a graceful energy *vs* quality scaling can be achieved for a specific set of computations. 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. Intuitively, if the energy budget is not sufficient for performing all the required adding operations, the last several numbers are least significant, and ignored. By doing so, the accuracy of the final sum can be sacrificed for energy cost. Such a transform may be more complicated for other applications. Moreover, the overhead of the transform should be relatively small. (The reader may refer to [171] for several examples pertinent to real applications.)

When image compression is involved in video surveillance applications, the aforementioned two system parameters — quantization level and VBS — can be used to adjust the quality of the delivered image in terms of both computation and communication energy costs. Intuitively, the quantization level affects the precision of the sampling, while the VBS tunes the fraction of pixels that are sampled by the compression algorithm. Either increasing the quantization level or decreasing the VBS leads to degraded image quality, and a smaller output size after compression.

Another challenge lies in the relationship between adaptive fidelity algorithms and other system knobs, including the three investigated in this book. For instance, it is not clear yet how the above algorithmic transform can be integrated with voltage scaling to further reduce energy costs.

Moreover, it is understood that the integration of adaptive fidelity algorithms with other system knobs should also be quite application-specific. Thus, identifying 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).

7.2.2 *A Broad View of Future Research*

7.2.2.1 *Mobile Sensor Nodes*

The work presented in the book is based on static networks, where the mobility of sensor nodes is not considered. Another trend in WSNs is the integration of mobile nodes into a static network [81]. The presence of mobile nodes changes the basic method for routing information across the network, implying a possible integration of existing results in mobile ad hoc networks with coding techniques for correlated information sources.

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 affect the decomposition of system energy costs. A large portion of the system energy is spent moving the mobile nodes. Thus, carefully scheduled node movements are crucial to maximize the system lifetime. Information availability and correlation among sources can be used to schedule the movements, so that energy is saved by avoiding visiting nodes with no new information.

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

There are also types of WSNs where all nodes are mobile, and need to cover the monitoring environment autonomously [81]. In this case, tradeoffs between the energy cost of the system and its coverage can be explored.

7.2.2.2 *Routing Diversity*

Routing diversity has been considered in various studies [66; 70; 97; 104; 126]. Two models have been considered, each with different focuses. When multi-path routing is considered at the routing layer, a simple disk model is often used to abstract the wireless transmission [66]. The key is to identify a set of disjoint paths by which to route the packets, so that a reliable packet delivery can be achieved despite unreliable wireless communication. Besides opportunistic data aggregation, existing techniques for multi-path routing have not formally addressed the issues of collaborative information processing with routing. Addressing these issues is challenging, due to the fact that multiple information flows from the same source become possible in the network. However, to perform data aggregation efficiently 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 [70; 97; 104; 126]. For example, the transmission of the same packet from multiple nodes can be coordinated to achieve the effect of multi-antenna transmission [97]. However, existing studies on routing diversity at the physical layer have not considered the collaboration between information processing and routing, which is an interesting and challenging topic for future work.

7.2.2.3 *Sleep Scheduling*

Sleep scheduling is also an important and widely used technology for low-traffic scenarios. It has been applied to various layers for different purposes, including reducing interference at the physical layer, optimizing packet scheduling at the MAC layer, enabling topology and routing control at the routing layer, and tuning sensing coverage at the application layer. It is not clear, however, how to integrate the techniques proposed in this book with the sleep scheduling of nodes. In the following, we discuss several relevant issues.

From the perspective of the entire system, the topology and routing control at the routing layer leads to a possible re-construction of the network infrastructure, including clustering and routing tree topology. Thus, there should be certain mechanisms to signal the changes in cluster members, or the structure of the routing tree. Subsequent operations to schedule the computation and communication tasks accordingly are also necessary, in light of such changes. For example, for the problem of information transportation over a tree substrate, our on-line protocol can also be used to handle changes in the tree structure based on its feedback mechanism. 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 changes in the cluster occur, with the hope of amortizing the cost of re-calculation in the case that such changes are rare. 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 policies designed at different layers can co-exist in the first place. It seems that coordination among these techniques is needed to provide an arbitration policy, so that decisions for the most significant functionality can override decisions for less critical needs. Considering the fact that there could potentially be 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.

This page is intentionally left blank

Bibliography

[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] N. Abramson. Development of the ALOHANET. *IEEE Trans. on Information Theory*, 31(2):119–123, Mar. 1985.

[3] J. Acimovic, B. Beferull-Lozano, and R. Cristescu. Adaptive distributed algorithms for power-efficient data gathering in sensor networks. In *IEEE International Symposium on Wireless Sensor Networks*, June 2005.

[4] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. on Information Theory*, 46(4):1204–1216, 2000.

[5] 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.

[6] 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.

[7] 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.

[8] ALERT systems. http://www.alertsystems.org.

[9] 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.

[10] 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.

[11] G. Asada, M. Dong, T. S. Lin, F. Newberg, G. J. Pottie, and W. J. Kaiser. Wireless integrated network sensor: Low power systems on a chip. In *ES-*

SCIRC '98, 1998.
[12] ATMEL ATmega128L Datasheet. http://www.atmel.com/dyn/resources/prod_documents/2467S.pdf.
[13] B. Awerbuch and Y. Azar. Buy-at-bulk network design. In *Annual Symposium on Foundations of Computer Science (FOCS)*, 1997.
[14] 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.
[15] 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.
[16] K. Barr and K. Asanović. Energy aware lossless data compression. In *First International Conference on Mobile Systems, Applicatiosn and Services*, May 2003.
[17] 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.
[18] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Annual Symposium on Foundations of Computer Science (FOCS)*, 1997.
[19] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Annual ACM Symposium on Theory of Computing (STOC)*, 1998.
[20] 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.
[21] T. Bell, M. Powell, J. Horlor, and R. Arnold. The Canterbury Corpus. http://www.cosc.canterbury.ac.nz.
[22] 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.
[23] K. Bharat-Kumar and J. Jaffe. Routing to multiple destination in computer networks. *IEEE Trans. on Computers*, 3(31):343–351, 1983.
[24] 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.
[25] G. Boone. Reality mining: Browsing reality with sensor networks.
[26] D. Bradinsky and D. Estrin. Rumor routing algorithm for sensor networks. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Sep. 2002.
[27] 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.
[28] K. Bult, A. Burstein, D. Chang, M. Dong, M. Fielding, E. Kruglick, J. Ho, F. Lin, T. H. Lin, W. J. Kaiser, H. Marcy, R. Mukai, P. Nelson, F. L.

Newburg, K. S. J. Pister, G. Pottie, H. Sanchez, O. M. Stafsudd, K. B. Tan, C. M. Ward, S. Xue, and J. Yao. Wireless integrated microsensors. In *Conference on Sensors and Systems (Sensors Expo)*, pages 33–38, Apr. 1996.

[29] P. Buonadonna, J. Chhabra, L. Krishnamurthy, and N. Kushalnagar. Heavy industry applications of sensornets. In *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2005. Poster.

[30] 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.

[31] 21 ideas for the 21st century. Business Week, pp. 78-167, Aug. 1999.

[32] R. Cardell-Oliver, K. Smettem, M. Kranz, and K. Mayer. A reactive soil moisture sensor network: Design and field evaluation. *International Journal of Distributed Sensor Networks*, 1(2):149–162, Apr. 2005.

[33] The Cooperative Engagement Capability. http://techdigest.jhuapl.edu/td1604/APLteam.pdf.

[34] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communication technology. In *ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, Apr. 2001.

[35] 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.

[36] 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.

[37] K. Chintalapudi, J. Caffrey, R. Govindan, E. Johnson, B. Krishnamachari, S. Masri, and G. Sukhatme. Networked active sensing of structures. In *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2005. Poster.

[38] K. K. Chintalapudi and R. Govindan. Localized edge detection in wireless sensor networks. In *IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*, Apr. 2003.

[39] C.-Y. Chong and S. P. Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8), Aug. 2003.

[40] CodeBlue project. http://www.eecs.harvard.edu/ mdw/proj/codeblue.

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

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

[43] CORIE Project. http://www.ccalmr.ogi.edu/CORIE/.

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

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

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

[47] R. Cristescu, B. Beferull-Lozano, and M. Vetterli. On network correlated

data gathering. In *IEEE InfoCom*, Mar. 2004.

[48] The CrossBow corporation. http://www.xbow.com.

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

[50] A. D'Costa and A. M. Sayeed. Collaborative signal processing for distributed classification in sensor networks. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks*, Apr. 2003.

[51] 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.

[52] A. P. Dempster. A generalization of bayesian inference. *Journal of the Royal Statistical Society*, 30:205–247, Apr. 1968.

[53] 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.

[54] In dust we trust. Economist, June, 12 2004.

[55] 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.

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

[57] 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.

[58] The Ember corporation. http://www.ember.com.

[59] 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.

[60] 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.

[61] The ExScal Project. http://cast.cse.ohio-state.edu/exscal.

[62] 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.

[63] 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.

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

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

[72] 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.

[73] 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.

[74] J. Hill. *System Architecture for Wireless Sensor Networks*. PhD thesis, University of California at Berkeley, 2003.

[75] J. Hill and D. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro.*, 22(6):12–24, Nov. 2002.

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

[77] 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.

[78] 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.

[79] 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.

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

[81] 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.

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

[83] A. T. Ihler, J. W. Fisher III, R. L. Moses, and A. S. Willsky. Nonparametric belief propagation for self-calibration in sensor networks. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks*, Apr. 2004.

[84] D. Ingraham, R. Beresford, K. kaluri, M. Ndoh, and K. Srinivasan. Wireless sensors: Oyster habitat monitoring in the bras d'or lakes. In *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2005. poster.

[85] 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.

[86] P. Ishwar, R. Puri, S. S. Pradhan, and K. Ramchandran. On compression for robust estimation in sensor networks. In *International Symposium on Information Theory*, July 2003.

[87] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Design Automation Conference (DAC)*, June 2004.

[88] D. B. Johnson and D. A. Maltz. *Mobile Computing*, chapter Dyanmic Source Routing in Ad Hoc Wireless Networks, pages 153–181. Kluwer Academic Publishers, 1996.

[89] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for widelife tracking: Design tradeoffs and early experiments with zebranet. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 96–107, Oct. 2002.

[90] J. Jubin and J. Tornow. The DARPA packet radio network protocols. *Proceedings of the IEEE*, 75(1):21–32, 1987.

[91] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transcation of the ASME journal of Basic Engineering*, 82:35–45, Mar. 1960.

[92] 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.

[93] 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.

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

[95] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *ACM/IEEE International Conference on Mobile Com-*

puting and Networking (MobiCom), Aug. 2000.

[96] A. Kashyap, L. A. Lastras-Montano, C. Xia, and Z. Liu. Distributed source coding in dense sensor networks. In *Data Compression Conference*, pages 13–22, Mar. 2005.

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

[98] 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.

[99] R. Kremens, J. Faulring, A. Gallagher, A. Seema, and A. Vodacek. Autonomous field-deployable wildland fire sensors. *International Journal of Wildland Fire*, 12:237–244, 2002.

[100] 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.

[101] B. Krishnamachari and S. Iyengar. Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor network. *IEEE Trans. on Computers*, 53(3):241–250, Mar. 2004.

[102] 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.

[103] 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.

[104] 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.

[105] F. L. Lewis. *Smart Environments: Technologies, Protocols, and Applications, edited by D. J. Cook and S. K. Das*, chapter Wireless Sensor Networks. John Wiley, 2004.

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

[107] 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.

[108] J. Liu, P. Cheung, L. Guibas, and F. Zhao. A dual-space approach to tracking and sensor management in wireless sensor networks. Technical Report P2002-10077, Palo Alto Research Center, Mar. 2002.

[109] 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.

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

[111] H. Luo and G. Pottie. Routing explicit side information for data gathering in wireless sensor networks. In *IEEE International Conference on Distributed*

Computing in Sensor Systems (DCOSS), June 2005.

[112] 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.

[113] R. C. Luo and M. G. Kay. *Data Fusion in Robotics and Machine Intelligence, edited by M. A. Abidi and R. C. Gonzalez*, chapter Data Fusion and Sensor Integration. Academic Press, 1992.

[114] J. P. Lynch. Decentralization of wireless monitroing and control technologies for smart civil structures. Technical Report 140, John A. Blume Earthquake Engineering Center, Stanford, 2002.

[115] 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.

[116] A. Mainwaring, J. Polastre, R. Szewczyk, D. Cullar, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Sep. 2002.

[117] 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.

[118] S. Martin, K. Flautner, T. Mudge, and D. Blaauw. Combined dynamic voltage scaling and adaptive body biasing for low power microprocessors under dynamic workloads. In *International Conference on Computer Aided Design*, Nov. 2002.

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

[120] UCLA MEDUSA platform. http://www.cens.ucla.edu/Project-Descriptions/Sensor_Node_Platforms.

[121] 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.

[122] MEMS technology application center. http://www.memsnet.org.

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

[124] 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.

[125] 10 emerging technologies that will change the world. MIT Technology Review, vol. 106, no. 1, pp. 33-49, Jan. 2003.

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

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

[128] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada. 1-v power supply highspeed digital circuit technology with multithreshold

voltage CMOS. *IEEE J. of Solid-State Circuits*, 30(8):847–854, 1995.

[129] 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.

[130] C. Neau and K. Roy. Optimal body bias selection for leakage improvement and process compensation over different technology generations. In *International Symposium on Low Power Electronics and Design*, Aug. 2003.

[131] D. Niculescu and B. Nath. Ad hoc positioning systems (APS). In *IEEE GlobeCom*, Nov. 2001.

[132] R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks*, Apr. 2003.

[133] Wireless sensor networks spread to new territory. The New York Times, July, 26 2004.

[134] PARSEC Project. http://pcl.cs.ucla.edu/projects/parsec.

[135] 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.

[136] PC104 testbed. http://www.isi.edu/scadds/pc104testbed.

[137] M. Perillo, Z. Ignjatovic, and W. Heinzelman. An energy conservation method for wireless sensor networks employing a blue noise spatial sampling. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks*, Apr. 2004.

[138] C. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, Feb. 1999.

[139] Picoradio: Berkeley wireless research center. http://bwrc.eecs.berkeley.edu/Research/Pico_Radio/.

[140] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks*, Apr. 2005.

[141] R. Pon, M. A. Batalin, V. Chen, A. Kansal, D. Liu, M. Rahimi, L. Shirachi, A. Somasundra, Y. Yu, M. Hansen, W. J. Kaiser, M. B. Srivastava, G. Sukhatme, and D. Estrin. Corodinated static and mobile sensing for environmental monitoring. In *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2005. Poster.

[142] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Comunications of the ACM*, 43(5):551–558, May 2000.

[143] 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.

[144] 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.

[145] 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.

[146] 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.

[147] 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.

[148] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. In *ACM SenSys*, Nov. 2003.

[149] R. Ramanathan and Rosales-Hail. Topology control of multihop wireless networks using transmit power adjustment. In *IEEE InfoCom*, pages 404–413, Mar. 2000.

[150] Redwoods go high tech: Researchers use wireless sensors to study california's state tree. UC Berkeley News, July 28, 2003.

[151] V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. *IEEE J. of Selected Areas in Communication (JSAC)*, 8(17):1333–1344, 1999.

[152] V. Sarkar. *Partitioning and Scheduling Programs for Execution on Multiprocessors*. The MIT Press, Cambridge, Massachusetts, 1989.

[153] M. Sartipi and F. Fekri. Source and channel coding in wireless sensor networks using LDPC codes. In *IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, Oct. 2004.

[154] A. Savvides, C. C. Han, and M. B. Srivastava. Dynamic fine grained localization in ad-hoc sensor networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, July 2001.

[155] A. Savvides, M. B. Srivastava, L. Girod, and D. Estrin. *Wireless Sensor Networks, edited by C. S. Raghavendra, K. M. Sivalingam and T. Znati*, chapter Localization in Sensor Networks. Springer, 2004.

[156] 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.

[157] 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.

[158] C. Schurgers, V. Raghunathan, and M. B. Srivastava. Modulation scaling for real-time energy aware packet scheduling. In *IEEE GlobeCom*, Nov. 2001.

[159] 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.

[160] 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.

[161] The Sensoria corporation. http://www.sensoria.com.

[162] S. Shakkottai, T. S. Rappaport, and P. C. Carlsson. Cross-layer design for wireless networks. *IEEE Wireless Communication Magazine*, 41(10):74–80, Oct. 2003.

[163] 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.

[164] Structural Health Monitoring Project. http://net-shm.usc.edu.

[165] C. Siaterlis and B. Maglaris. Towards multisensor data fusion for DoS detection. In *ACM Symposium on Applied Computing*, pages 439–446, Mar. 2004.

[166] M. Singh and V. K. Prasanna. System level energy tradeoffs for collaborative computation in wireless networks. In *IEEE IMPACCT Workshop*, May 2002.

[167] 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.

[168] 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.

[169] A. Sinha and A. P. Chandrakasan. Dynamic power management in wireless sensor networks. *IEEE Design and Test of Cmputers*, 18(2):62–74, 2001.

[170] 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.

[171] 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.

[172] D. Slepian and J. Wolf. Noiseless coding of correlated information sources. *IEEE Trans. on Information Theory*, 19(4):471–480, 1973.

[173] The Smart Dust Project. http://robotics.eecs.berkeley.edu/ pister/SmartDust.

[174] SRC SoC Design Challenge. http://www.src.org/SoC_contest/.

[175] H. Sohn and C. R. Farrar. Damage diagnosis using time series analysis of vibration signals. *Smart Materials and Structures*, 10(3):446–451, 2001.

[176] K. Sohrabi and G. J. Pottie. Performance of a novel self-organization protocol for wireless ad hoc sensor networks. In *IEEE Vehicular Technology Conference (VTC)*, 1999.

[177] 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.

[178] E. G. Straser and A. S. Kiremidjian. A modular wireless damage monitoring system for structures. Technical Report 128, John A. Blume Earthquake

Engineering Center, Stanford, 1998.
[179] 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.
[180] H. Takahashi and A. Matsuyama. An approximate solution for the steiner problem in graphs. *Mach. Japonica*, 24(6):573–577, 1980.
[181] C. Tang and C. S. Raghavendra. Bitplane coding for correlation exploitation in wireless sensor networks. In *IEEE International Conference on Communications (ICC)*, May 2005.
[182] C. N. Taylor and S. Dey. Adaptive image compression for wireless multimedia communication. In *IEEE International Conference on Communications (ICC)*, June 2001.
[183] 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.
[184] The TinyDB Project. http://telegraph.cs.berkeley.edu/tinydb/.
[185] TinyOS website. http://www.tinyos.net/.
[186] 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.
[187] MIT μAPMS Project. http://www-mtl.mit.edu/researchgroups/icsystems/uamps.
[188] 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.
[189] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *ACM SenSys*, Nov. 2003.
[190] P. von Rickenbach and R. Wattenhofer. Gathering correlated data in sensor networks. In *ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, Oct. 2004.
[191] 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.
[192] 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.
[193] E. Welsh, W. Fish, and P. Frantz. GNOMES: A testbed for low-power heterogeneous wireless sensor networks. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2003.
[194] K. Whitehouse and D. Culler. Macro-calibration in sensor/actuator networks. *ACM Mobile Networks and Applications (MONET)*, 8(4):463–472, Aug. 2003. Special Issue on Wireless Sensor Networks.
[195] R. Willett, A. Martin, and R. Nowak. Backcasting: Adaptive sampling for sensor networks. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks*, Apr. 2004.

[196] H. P. Williams. *Model Building in Mathematical Programming.* John Wiley & Sons, Ltd, 1999.

[197] The WINS Project, Rockwell Science Center. http://wins.rsc.rockwell.com.

[198] J.-J. Xiao, S. Cui, Z.-Q. Luo, and A. Goldsmith. Joint estimation in sensor networks under energy constraints. In *IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, Oct. 2004.

[199] N. Xu. http://enl.usc.edu/ ningxu/papers/survey.pdf.

[200] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *ACM SenSys*, Nov. 2004.

[201] 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.

[202] 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.

[203] 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.

[204] 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.

[205] Y. Yao and G. B. Giannakis. Energy-efficient scheduling for wireless sensor networks. *IEEE Trans. on Computers*, 53(8):1333–1342, 2005.

[206] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *IEEE InfoCom*, June 2002.

[207] 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.

[208] 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.

[209] Y. Yu and V. K. Prasanna. Energy-balanced multi-hop packet transmission in wireless sensor networks. In *IEEE GlobeCom*, Dec. 2003.

[210] 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.

[211] 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.

[212] Y. Zhang and L. Cheng. Cross-layer optimization for sensor networks. In *New York Metro Area Networking Workshop*, Sep. 2003.

[213] Y. Zhang, X. Hu, and D. Z. Chen. Task scheduling and voltage selection for energy minimization. In *Design Automation Conference (DAC)*, 2002.

[214] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collab-

oration for tracking applications. *IEEE Signal Processing Magazine*, Mar. 2002.

[215] 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.

[216] C. Z. Zhou and B. Krishnamachari. Localized topology generation mechanisms for self-configuring sensor networks. In *IEEE GlobeCom*, Dec. 2003.

[217] 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 5.1: $\Rightarrow$ 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. Note that this condition does not mean the length of all paths containing V_i equals Γ, as demonstrated by the second example in the comments after the proof.

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 is the minimization of $f(\vec{\tau})$, subject to $A\vec{\tau} = \vec{\Gamma}$. Based on the first-order necessary condition for linearly constrained problems [119], 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 adding 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 5.1 in three interesting examples.

In the first example, consider a linear array of v sensor nodes. Assume that the latency constraint is tight enough so that $\sum_1^{v-1} m_i >\geq \Gamma$. Let $\vec{\tau^*}$ denote the optimal schedule. It is understood from Lagrangian relaxation that for the optimal solution, the first derivatives of the energy function on all edges shall be equal to each other and $\sum_1^{v-1} \tau_i^* = \Gamma$. Hence, Lemma 5.1 holds.

In the second example, consider an aggregation tree $T = (\mathbb{V}, \mathbb{L})$ where $\mathbb{V} = \{V_1, V_2, V_3, V_4\}$ and $\mathbb{L} = \{(1,3),(2,3),(3,4)\}$. Let $\vec{\tau^*}$ denote the optimal schedule of such a problem. Assume that $m_2 + m_3 \geq \Gamma$ and m_1 is far smaller than m_2, m_3, and Γ, such that we have $\tau_1^* = m_1 < \tau_2^*$ in the optimal schedule. In this case, we have $\dot{w}_1(\tau_1^*) = 0$. Moreover, since the transmission time over $(1,3)$ is dominated by the transmission time over $(2,3)$, it suffices to find an optimal schedule for the sub-problem that consists of edges $(2,3), (3,4)$ only. From the first example, we have $\dot{w}_2(\tau_2^*) = \dot{w}_3(\tau_3^*)$ and $\tau_2^* + \tau_3^* = \Gamma$ for the optimal solution, which follows from Lemma 5.1. Also note that in this case, the length of path $\{V_1, V_3, V_4\}$ is less than Γ.

In the third 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 $i = 1, \ldots, n-1$. In such a case, we have $\dot{w}_i(\tau_i^*) = 0$ for $i = 1, \ldots, n-1$, which again respects Lemma 5.1.

More importantly, we note the following fact in the optimal solution for the third 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 A.1 and Theorem 5.1 such that θ_i must equal $\max_{(j,i)\in \mathbb{L}}(\theta_j + \tau^* j)$.

Corollary A.1 *Consider an optimal schedule, $\vec{\tau^*}$, of OPTP; the following hold:*

(1) Suppose $\tau_i^* = m_i$ for some $V_i \in \mathbb{V}$, we have $\tau_j^* = m_j$ for all sensor nodes in T_i.

(2) Suppose $\tau_i^* < m_i$ for some $V_i \in \mathbb{V}$, we have $\tau_j^* < m_j$ for all ancestors of V_i.

Proof of Corollary A.1:

(1) Since $\tau_i^* = m_i$ implies $\sum_{(j,i)\in\mathbb{L}} \dot{w}_j(\tau_j^*) = \dot{w}_i(\tau_i^*) = 0$, we have $\tau_j^* = m_j$ for all children of V_i. Thus, 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 $\dot{w}_j(\tau_j^*) \leq \dot{w}_i(\tau_i^*) < 0$; therefore $\tau_j^* < m_j$. □

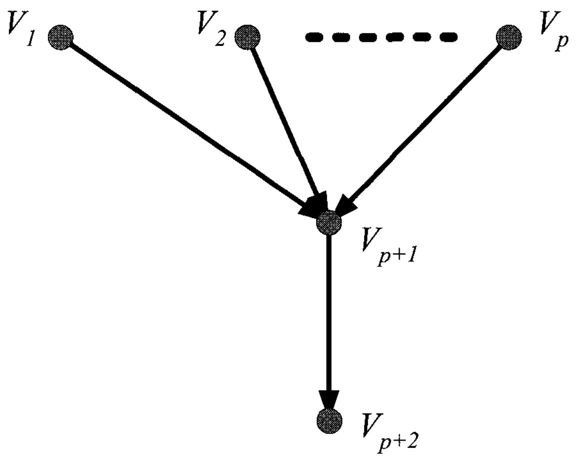


Fig. A.1 A problem instance of 2-Lev-OPTP

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 $\mathcal{C} = \{V_1, \ldots, V_p\}$ denoting its children. Assume that for any $V_i \in \mathcal{C}$, a packet is ready to transmit at time s_i and V_{p+2} must receive aggregated information from V_{p+1} by time t. We first prove the following lemma:

Lemma A.1 *Let $\vec{\tau}^* = \{\tau_1^*, \ldots, \tau_{p+1}^*\}$ denote an optimal schedule to the 2-Lev-OPTP problem as defined above; the following hold:*

(1) The schedule $\vec{\tau}^*$ is unique.
(2) Let s_{p+1} denote the start time of packet transmission from V_{p+1} to V_{p+2} in the optimal schedule, i.e., $s_{p+1} = \max_{V_i\in\mathcal{C}}(s_i+\tau_i^*)$. Then s_{p+1} never decreases when (a) some s_i's, $V_i \in \mathcal{C}$, increase, holding t fixed; or (b) t

increases, holding s_i's fixed, for all $V_i \in \mathcal{C}$; or (c) both some s_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, s_{p+1} increases in case (a) if for any $V_i \in \mathcal{D}$, we have $s_{p+1} - s_i \leq m_i$; or in case (b) we have $t - s_{p+1} < m_{p+1}$; or in case (c) we have either of the previous two conditions hold.

(3) *s_{p+1} never increases when (a) some (≥ 1) s_i's, $V_i \in \mathcal{C}$, decrease, holding t fixed; or (b) t decreases, holding s_i's fixed, for all $V_i \in \mathcal{C}$; or (c) both some s_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, s_{p+1} decreases in case (a) if for any $V_i \in \mathcal{D}'$, we have $s_{p+1} - s_i < m_i$; or in case (b) we have $t - s_{p+1} \leq m_{p+1}$; or in case (c) we have either of the previous two conditions hold.

Proof of Lemma A.1:

(1) The uniqueness of the optimal solution follows the strict convexity of the energy functions.

(2) From Lemma 5.1, 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 $V_a \in \mathcal{C}$, s_a increases to s'_a. Let $\hat{s}_{p+1}$ 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 $s_{p+1} = s_a + \tau_a^*$. Otherwise, a schedule with less energy dissipation than $\vec{\tau}^*$ can be constructed by increasing τ_a^* by $\delta \leq \min\{s_{p+1} - (s_a + \tau_a^*), m_a - \tau_a^*\}$ without affecting τ_{p+1}^* or violating the latency constraint. Similarly, we have $\tau_{p+1}^* \leq m_{p+1}$ and $s_{p+1} + \tau_{p+1}^* = t$.

Now we consider the problem resulting from increasing s_a to s'_a. Suppose that in the new packet schedule, we still enforce the transmission by V_{p+1} to start at s_{p+1}, we have:

$$\dot{w}_a(\tau_a^* - (s'_a - s_a)) + \sum_{V_i \in \mathcal{C} \wedge i \neq a} \dot{w}_i(\tau_i^*) < \sum_{V_i \in \mathcal{C}} \dot{w}_i(\tau_i^*)$$
$$= \dot{w}_{p+1}(\tau_{p+1}^*) \, . \tag{A.4}$$

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 $s_{p+1} + (s'_a - s_a)$ in the new schedule. Since $\tau^*_{p+1} \leq m_{p+1}$, we have:

$$\begin{aligned}
& \dot{w}_a(\tau^*_a) + \sum_{V_i \in \mathcal{C} \wedge i \neq a} \dot{w}_i(\min\{\tau^*_i + (s'_a - s_a), m_i\} \\
& \geq \sum_{V_i \in \mathcal{C}} \dot{w}_i(\tau^*_i) \\
& = \dot{w}_{p+1}(\tau^*_{p+1}) \\
& > \dot{w}_{p+1}(\tau^*_{p+1} - (s'_a - s_a)) \ . \qquad \text{(A.5)}
\end{aligned}$$

Equations A.4 and A.5, and the uniqueness of the optimal schedule, imply that $s_{p+1} < \hat{s}_{p+1} < s_{p+1} + (s'_a - s_a)$.
Case (ii): Suppose that in schedule $\vec{\tau^*}$, $\tau^*_a = m_a$; hence, we have $s_{p+1} - s_a \geq m_a$. If $s_{p+1} - s'_a < m_a$, a similar analysis as in Case (i) can be carried out to show that $(s_{p+1} < \hat{s}_{p+1} < s_{p+1} + (s'_a - s_a)$. Otherwise, $\vec{\tau^*}$ remains an optimal schedule, implying that $\hat{s}_{p+1} = s_{p+1}$.
In the case when multiple s_i's ($V_i \in \mathcal{C}$) increase, it can handled by increasing these s_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 s_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 5.1.
Proof of Theorem 5.1:

(1) Recall that EMR-Algo works in iterations: for each iteration k, the algorithm determines s^k_i by decreasing i from $v-1$ to $M+1$. Since the EMR-Algo initializes $s_i = 0$ for $i = 1, \ldots, v-1$, it follows that $s^0_i \leq s^1_i$ for each $i = 1, \ldots, v-1$. Suppose that $i' > 1$ and $k' > 1$ is the first time that there is a violation; that is, $s^{k'}_{i'} > s^{k'+1}_{i'}$.
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 $s^{k'}_p \leq s^{k'+1}_p$, and $s^{k'-1}_i \leq s^{k'}_i$, for each $V_i \in \mathcal{C}$.
From line 8 in EMR-Algo, the time stamps $s^{k'}_p$ and $s^{k'-1}_i$'s actually give the boundaries within which EMR-Algo determines $s^{k'}_{i'}$. Similarly, the time stamps $s^{k'+1}_p$ and $s^{k'}_i$'s give the boundaries within which EMR-Algo determines $s^{k'+1}_{i'}$. From part (2) of Lemma A.1, we have $s^{k'}_{i'} \leq$

$s_{i'}^{k'+1}$. 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 $k' \geq 1$ is 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 A.1 again leads to the contradiction that $s_{i'}^{k'} \leq s_{i'}^*$ and proves property (2).

(3) We prove by contradiction, and hence assume that $j = \max\{i : s_i^\infty < s_i^*\}$. Let V_p denote the parent of V_j and V_g denote the parent of V_p. We have $s_p^\infty = s_p^*$ and $s_g^\infty = s_g^*$. 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 $s_j^\infty < s_j^*$ and $s_i^\infty \leq s_i^*$, for each $V_i \in \mathcal{C} \wedge i \neq j$. Suppose that we run EMR-Algo for one more pass, and let $\hat{s}_p$ denote the resulting start time for the transmission from V_p to V_g. From part (3) of Lemma A.1, we have $s_p^\infty - (s_j^* - s_j^\infty) < \hat{s}_p < s_p^\infty = s_p^*$, 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 A.1, we have $\tau_i^* = m_i$ for any $V_i \in T_j$. Moreover, we have $s_p^\infty - s_j^\infty > s_p^* - s_j^* = \tau_j^* = m_j$. Obviously, we shall have $\tau_j^\infty = m_j$. Again from part (1) of Corollary A.1, we have $\tau_i^\infty = m_i$ for any $V_i \in T_j$. Based on the definition of s_j^∞ and s_j^*, we obtain the contradiction that $s_j^\infty = s_j^*$.
□

Appendix B

Performance Bound of SPT and MST for TDG problem with grid deployment

Proof.

We first lower bound the optimal cost. From (6.8a) and the fact $\gamma^* = (2r-1)L_1^2 = \Omega(r)$, we have

$$\begin{aligned} \lim_{\gamma \to \gamma^*} \varepsilon_{LB} &= \Omega(r\sqrt{\gamma r}) = \Omega(r^2) \quad \text{and} \\ \lim_{\gamma \to 0} \varepsilon_{LB} &= \Omega(r^2 L_r) = \Omega(r H_r) \ . \end{aligned}$$

Equation (6.9) directly follows from Corollary 6.1. To prove (6.10), we bound the cost of SPT based on (6.6b). From the condition in (6.6b), we have $\frac{\gamma}{L_r} \leq rL_r = H_r$ and thus

$$\begin{aligned} \lim_{\gamma \to 0} \varepsilon_{SPT} &\leq \frac{r(r-1)L_1}{2} + 2rH_r + \sum_{j=1}^{r-1} H_j \\ &= O(r^2) + O(rH_r) \ . \end{aligned}$$

Hence, (6.10) follows.

From the condition in (6.7a), $\gamma \to \gamma^*$ implies $i \to i^*$. Also, we have

$\frac{\gamma}{f^*} + f^*(q-i) \leq \frac{\gamma}{L_{i-1}} + L_{i-1}(q-i) \leq O(L_{i-1}(q-i))$. The second inequality comes from the boundary of γ in (6.7a). Since $\gamma^* = (2r-1)L_1^2$, we have

$$\begin{aligned}
\lim_{\gamma \to \gamma^*} \varepsilon_{MST} &= O(\sqrt{r} \sum_{j=2r-1}^{q-i^*-1} \sqrt{j}) + \sum_{j=1}^{i^*-1} H_j \\
&\quad + O(i^*(q-i^*)L_{i^*-1}) \\
&= O(r^2) + \sum_{j=1}^{i^*-1} H_j + O((q-i^*)H_{i^*-1}) \\
&= O(r^2) + O(rH_r) \; .
\end{aligned}$$

Thus,

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

Lastly, we bound the cost of $\lim_{\gamma \to 0} \varepsilon_{MST}$. From the condition in (6.7b), we have $\frac{\gamma}{L_r} \leq 2rL_r$. Hence,

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

and (6.12) follows. $\square$

Index

Information Processing and Routing in Wireless Sensor Networks

This book presents state-of-the-art cross-layer optimization techniques for energy-efficient information processing and routing in wireless sensor networks. Besides providing a survey on this important research area, three specific topics are discussed in detail — information processing in a collocated cluster, information transport over a tree substrate, and information routing for computationally intensive applications. The book covers several important system knobs for cross-layer optimization, including voltage scaling, rate adaptation, and tunable compression. By exploring tradeoffs of energy versus latency and computation versus communication using these knobs, significant energy conservation is achieved.

Key Features

- Presents state-of-the-art cross-layer optimization techniques in wireless sensor networks
- Covers end-to-end framework for information processing and routing
- Accessible to students, yet of interest to experts

World Scientific
www.worldscientific.com
6288 hc

ISBN 981-270-146-X
9 789812 701466