Energy Aware Task Allocation Algorithms for Wireless Sensor Networks

Dem Fachbereich Physik und Elektrotechnik
Der Universität Bremen

Dissertation zur Erlangung des akademischen Grades eines
Doktor-Ingenieur (Dr.-Ing.)

von

M.Sc. Wanli Yu

Referent:	Prof. Dr.-Ing. Alberto García-Ortiz
Korreferent:	Prof. Dr.-Ing. Karl-Ludwig Krieger
Eingereicht am:	01.02.2018
Tag des Promotionskolloquiums:	

Acknowledgments

I would like to take this opportunity to thank all those who made this work complete.

First and foremost, I would like to warmly thank Prof. Dr.-Ing. Alberto García-Ortiz. He is such a wonderful doctoral supervisor who guides and inspires me a lot with his infinite patience, careful guidance and significant advice. The enthusiasm for research, the way of thinking and the kindness to people I have learned from him are really treasures for my future career.

I would also like to express my appreciation to Prof. Dr.-Ing. Karl-Ludwig Krieger, who accepts to review my dissertation. He is one of the most professional and insightful professors I have ever met. Many thanks to Prof. Dr. Anna Förster and Prof. Dr.-Ing. Udo Frese for discussing and understanding my work and the kind participation in my dissertation defense.

During the past four years, I have spent an enjoyable time in IDS research group. Special thanks to Kerstin Janssen, who has spared no effort with the administration work, the language problems and daily life supports. Sincere gratitude to my colleagues Lennart Bamberg, Amir Najafi, Ardalan Najafi, Ayad Dalloo and Peter Lutzen, who have granted me many helps. Also, I would like to say thank you to colleagues Daniel Gregorek, Wolfgang Büter, Christof Osewold, Parham Haririan, Maike Schröder. I really enjoy every moment with them.

In particular, I gratefully acknowledge the financial supports from China Scholarship Council and Prof. Dr.-Ing. Alberto García-Ortiz.

At last, I am deeply grateful to my parents. They are always my backup and provides unreserved support whenever I need. To my fiancée Dr.-Ing. Yanqiu Huang, I am so grateful to meet you in my life and always have you by my side.

Abstract

Complex wireless sensor network (WSN) applications, such as those in Internet of things or in-network processing, are pushing the requirements of energy efficiency and long-term operation of the network drastically. Energy aware task allocation becomes crucial to extend the network lifetime, by efficiently distributing the tasks of applications among sensor nodes. Although task allocation has been deeply studied in wired systems, the resulting approaches are insufficient for WSNs due to limited battery resources and computing capability of WSN nodes, as well as the special wireless communication.

This work focuses on designing energy aware task allocation algorithms to extend the network lifetime of WSNs. More precisely, this work firstly proposes a centralized static task allocation algorithm (CSTA) for cluster based WSNs. Since a WSN application can be modeled by a directed acyclic graph (DAG), the task allocation problem is formulated as partitioning the modeled DAG graph into two subgraphs: one for the slave node and the other for the master node. By using a binary vector variable to represent the partition cut, CSTA formulates the problem of maximizing network lifetime as a binary integer linear programming (BILP) problem. It provides one fixed time invariant partition cut (task allocation solution) for each slave node to balance the workload distribution of tasks. Moreover, motivated by the fact that using multiple partition cuts can achieve more balanced workload distribution, this work extends CSTA to a centralized dynamic task allocation algorithm, CDTA. By using a probability vector variable to stand for partition cuts with different weights, CDTA formulates the dynamic task allocation problem as a linear programing (LP) problem. Due to the high complexity of centralized algorithms, this work further proposes a very lightweight distributed optimal on-line task allocation algorithm (DOOTA). Through an indepth analysis, it proves that the optimal task allocation solution consists of at most two partition cuts for each slave nodes. Based on this analysis, DOOTA enables each slave node to calculate its own optimal task allocation solution by negotiating with the master node with a very short time. These contributions significantly improve the application performance for WSNs, but also for other domains, e.g, mobile edge/fog computing.

Furthermore, the proposed task allocation algorithms are extended for different task scenarios and network structures, i.e., applications with conditional tasks, joint local and global appli-

cations and multi-hop mesh network. Given a condition triggered application, it is modeled by a DAG graph with conditional branches. This conditional DAG is further decomposed into multiple stationary DAG graphs without conditional branches according to the satisfaction probability of each condition. Based on this modeling, a static and a dynamic condition triggered task allocation algorithms (SCTTA and DCTTA) are proposed by considering the multiple stationary DAG simultaneously. Targeting the joint local and global applications, this work designs a static and a dynamic joint task allocation algorithms, SJTA and DJTA, based on BILP and LP, respectively. The modeling of local task allocation problem does not change, while the global task allocation problem is modeled by dividing the global DAG graph into different subgraphs mapping to the slave and master nodes. Besides the extensions for different task scenarios, this work presents a dynamic task allocation algorithm for multi-hop mesh networks (DTA-mhop) as well. The corresponding task allocation problem is modeled by dividing the DAG graph of each sensor node into multiple subgraphs mapping to itself, the routing and sink nodes. By using the summation of assigned tasks for each node, DTA-mhop formulate the lifetime maximization as a LP problem.

The proposed task allocation algorithms are firstly evaluated using simulations and real WSN applications, in terms of network lifetime increase and algorithm runtime. In order to investigate the algorithm's performance in realistic scenarios, the CSTA, CDTA and DOOTA algorithms are implemented in a real WSN based on the OpenMote platform. Both the simulation and implementation results show that the network lifetime can be dramatically extended. Remarkably, the network lifetime improvements are more significant for addressing complex applications. The proposed task allocation algorithms are therefore suitable for WSNs, and they can also be easily adapted to other wireless domains.

Kurzfassung

Komplexe drahtlose Sensor-Netzwerke (engl. Wireless-Sensor-Networks, WSNs), wie sie zum Beispiel in Industrie-4.0-Anwendungen zu finden sind, erfordern eine maximierte Energieeffizienz um eine lange Betriebsdauer der Netzwerke zu gewährleisten. Eine vielversprechende Technik zur Maximierung der Betriebsdauer ist die Task-Allocation. Hierbei werden den Prozessen Sensorknoten so zugewiesen, dass alle Sensorknoten in etwa gleichstark ausgelastet werden, um die Lebensdauer/Energieeffizienz zu steigern. Die Grundidee der Task-Allocation wurde bereits intensiv für festverdrahtete Systeme erforscht. Dennoch sind die existierenden Task-Allocations-Algorithmen für drahtlose Systeme aufgrund ihrer geringeren Rechenleistung, der limitierten Batteriekapazitäten und des unterschiedlichen physikalischen Übertragungsmediums ineffizient.

Die vorliegende Arbeit befasst sich mit dem Design von Task-Allocations-Algorithmen zur Maximierung der Energieeffizienz von WSNs, mit dem Ziel, deren Betriebsdauer zu maximieren. Präziser formuliert präsentiert diese Arbeit einen zentralisierten statischen Task-Allocations-Algorithmus (engl. Centralized-Static-Task-Allocation, CSTA) für Clusterbasierte WSNs. Da WSN-Anwendungen mittels eines gerichteten Graph ohne Zyklus (engl. Directed-Acyclic-Graph, DAG) modelliert werden können, lässt sich das Task-Allocations-Problem als eine Partitionierung des DAGs in zwei Teilgraphen formulieren: Einer für die Slave-Nodes und ein weiterer für den Master-Node. Ein binärer Vektor wird genutzt um die Partitionierung auszudrücken. Dadurch kann die Maximierung der Energieeffizienz mittels CSTA-Verfahren als ein Problem der binären linearen Programmierung (engl. Binary-Integer-Linear- Programming, BILP) formuliert werden. Das Verfahren resultiert in einer fixen, zeit-invarianten Lösung des Task-Allocations-Problems (Partitionierungs-Schnitt) für jeden Slave-Node, welche zu einer gleichmäßigen Auslastung der Sensorknoten, und folglich zu einer Maximierung der Energieeffizienz, führen.

Motiviert durch den Fakt das mehrere Partitionierungsschnitte die Energieeffizienz weiter steigern können, erweitert die vorliegende Arbeit das CSTA-Verfahren ferner zu einem zentralisierten, dynamischen Task-Allocations-Verfahren (CDTA). Mithilfe eines Vektors von Wahrscheinlichkeiten, der verschieden gewichtete Partitionierungsschnitte präsentiert, formuliert das CDTA-

Verfahren die Task-Allocation als lineares Programmierungsproblem. Aufgrund der hohen Komplexität von zentralisierten Verfahren stellt diese Arbeit ebenfalls einen dezentralisierten optimalen online Task-Allocations-Algorithmus (DOOTA) vor. Dieses Verfahren weist eine deutlich geringere Komplexität auf. Eine detaillierte Analyse zeigt, dass die optimale Task-Allocations-Lösung maximal zwei Partitionierungsschnitte für jeden Slave-Node beinhaltet. Basierend auf dieser Analyse ermöglicht das DOOTA-Verfahren es, dass jeder Slave-Node, durch Kommunikation mit dem Master-Node, innerhalb kürzester Zeit seine optimale Task-Allocation individuell berechnet, wodurch die Anwendungsperformance von WSNs signifikant verbessert wird. Die Anwendungsszenarien sind jedoch nicht auf WSNs beschränkt, da die vorgestellten Verfahren auch für andere Anwendungsbereiche der drahtlosen Kommunikation verwendet werden können (z.B. Mobile-Edge-Computing).

Darüber hinaus werden die vorgeschlagenen Task-Allocations-Algorithmen für verschiedene Anwendungsszenarien und Netzwerkstrukturen erweitert, wie zum Beispiel bedingte Tasks, gemeinsame lokale/ globale Netzwerkanwendungen und Multi-Hop-Netzwerke. Eine bedingt gestartete Applikation wird mittels DAG mit konditionellen Übergängen modelliert. Dieser bedingte DAG wird, in Abhängigkeit von den Bedingungen, weiter unterteilt in mehrere statische DAG. Basierend auf dieser Modellierung wird ein statischer (SCTTA) und ein dynamischer (DCTTA) Task-Allocations-Algorithmus für bedingt ausgeführte Tasks vorgestellt. Hierbei werden die statischen DAG zeitgleich berücksichtigt. Für gemeinsame lokale und globale Anwendungen werden ein statischer und ein dynamischer Algorithmus vorgestellt: SJTA basierend auf binärer ganzzahliger linearer Programmierung, und DJTA basierend auf linearer Programmierung. Die Modellierung der lokalen Task-Allocation bleibt unverändert, während die globale Task-Allocation mittels einer Unterteilung des globalen Graphens in Teilgraphen modelliert wird.

Neben den bereits genannten Erweiterungen beinhaltet diese Arbeit ebenfalls ein dynamisches Task-Allocations-Verfahren für Multihop-Netzwerke (DTA-mhop). Das zugehörige Task-Allocations-Problem wird modelliert durch eine Unterteilung des DAG von jedem Sensorknoten in mehrere Teilgraphen, welche sich selber, das Routing-Verfahren und die Sink-Nodes abbilden. Durch die Summe aus den zugewiesenen Tasks für jeden Knoten/Node, formuliert das DTA-mhop-Verfahren die geforderte Maximierung der Betriebszeit als ein Problem der linearen Programmierung. Die präsentierten Task-Allocations-Verfahren werden, zunächst anhand von Simulationen für reale WSN Applikationen, hinsichtlich Erhöhung der Betriebszeit und Laufzeit der Algorithmen evaluiert. Um die Performance der Verfahren für reale Netzwerke zu untersuchen, werden das CSTA-, das CDTA- und das DOOTA-Verfahren in einem realen WSN, basierend auf der OpenMote-Plattform, implementiert. Die Ergebnisse der simulativen

wie der experimentellen Evaluation zeigen, dass die vorgestellten Verfahren die Betriebszeiten von WSNs drastisch erhöhen können. Hierbei ist anzumerken, das für komplexe Applikationen die Erhöhung der Betriebszeit am signifikantesten ist. Daher eignen sich die präsentierten Verfahren hervorragend für moderne WSNs. Zudem können die Verfahren einfach an andere Anwendungsbereiche der Drahtloskommunikation angepasst werden.

Contents

1 Introduction **1**
1.1 Motivation 2
1.2 Main Contribution 2
1.3 Publications 5
1.4 Dissertation Outline 6

2 Background and Related Works **9**
2.1 Introduction 9
2.2 Wireless Sensor Networks 10
2.2.1 Typical WSN Topology Structures 11
2.2.2 WSN Applications 13
2.3 Related Works for Task Allocation in WSNs 14
2.4 Summary 18

3 System Models Used for Task Allocation Problem **21**
3.1 Introduction 21
3.2 Modeling WSN Applications 22
3.2.1 Graphs 22
3.2.2 WSN Application Models 23
3.3 Modeling the Energy Consumptions of WSN nodes 24
3.3.1 Cost Functions 25
3.3.2 A Detailed Communication Energy Cost Model for WSN Nodes 26
3.3.3 Validation of the Proposed Communication Energy Cost Model 29
3.4 Evaluation Metrics for Task Allocation Algorithms 32
3.5 Summary 35

4 Task Allocation Algorithms for Cluster-Based WSNs **37**
4.1 Introduction 38
4.2 System Model for CSTA, CDTA and DOOTA 38

4.3 CSTA: Centralized Static Task Allocation Algorithm for Cluster-Based WSNs . 39
4.3.1 Modeling the Static Task Allocation Problem 39
4.3.2 CSTA Algorithm 40
4.4 CDTA: Centralized Dynamic Task Allocation Algorithm for Cluster-Based WSNs 44
4.4.1 Modeling the Dynamic Task Allocation Problem 45
4.4.2 CDTA Algorithm 45
4.5 DOOTA: Distributed Optimal On-line Task Allocation Algorithm for Cluster-Based WSNs 48
4.5.1 Analysis of the Optimal Task Allocation Solution 48
4.5.2 DOOTA Algorithm 50
4.6 Simulation Results 56
4.6.1 Simulation Setup 56
4.6.2 Evaluations of CSTA, CDTA and DOOTA for Symmetric WSNs 57
4.6.3 Evaluations of CSTA, CDTA and DOOTA for Asymmetric WSNs 63
4.7 Summary 67

5 Extensions for Different Task Scenarios and Network Structures 71
5.1 Introduction 71
5.2 Extensions of Task Allocation for Different Task Scenarios 73
5.2.1 Task Allocation Algorithms for Condition Triggered Tasks: SCTTA and DCTTA 73
5.2.2 Task Allocation Algorithms for Joint Local and Global Tasks: SJTA and DJTA 79
5.3 Extensions of Task Allocation for Multi-hop Mesh WSNs: DTA-mhop 86
5.3.1 Modeling the Task Allocation for Multi-hop Mesh WSNs 86
5.3.2 DTA-mhop Algorithm 87
5.4 Simulation Results 92
5.4.1 Simulation Setup 92
5.4.2 Evaluation of SCTTA and DCTTA Algorithms 93
5.4.3 Evaluation of SJTA and DJTA Algorithms 97
5.4.4 Evaluation of DTA-mhop Algorithm 100
5.5 Summary 104

6 Experimental Results 107
6.1 Introduction 107

6.2 Estimation Using Real WSN Applications 108
6.2.1 Experiment Setup 108
6.2.2 Experimental Results 110
6.3 Hardware Implementation Using OpenMote-CC2538 Platform 116
6.3.1 OpenMote-CC2538 116
6.3.2 Hardware Implementation Results 117
6.4 Summary 122

7 Conclusion and Outlook 125
7.1 Summary of Contributions 125
7.2 Future Works 126

8 Appendix 129
8.1 Current Profile of ContikiMAC Protocol Overhead 129

1 Introduction

1.1 Motivation **2**

1.2 Main Contribution **2**

1.3 Publications **5**

1.4 Dissertation Outline **6**

In past decades, wireless sensor networks (WSNs) have received tremendous attention from both industrial and academic communities all over the world. A WSN typically consists of spatially distributed and mutually communicated sensor nodes that are deployed to monitor the physical or environmental phenomena. The small, rugged, inexpensive and low powered sensor nodes enable WSNs to be one of the fundamental technologies of Internet of Things (IoT) [1, 2]. This key technology has been applied to a wide range of applications and services, such as environment monitoring [3, 4], enhanced industrial control [5, 6], remote health care [7, 8], intelligent logistic [9, 10], smart home [11, 12], etc. It is envisioned that WSNs will revolutionize the way we live, work and interact with the physical world [13].

The WSN sensor nodes are typically powered by battery energy. In many WSN applications, it is very hard and sometimes even impossible to recharge or replace the dying sensor nodes due to the harsh physical environment and the large quantities. This will lead to fragmentations of the whole network and loss of potentially important information. Thus, how to achieve the energy efficiency and to extend the network lifetime is an extremely crucial issue for most WSN applications.

WSN applications consist of various kinds of tasks like sensing, processing, transmitting and receiving. The allocation of the tasks has a strong effect on the energy consumption. This dissertation aims to maximize the network lifetime by designing suitable task allocation algorithms.

1.1 Motivation

The energy related activities of a sensor node typically involves sensing, processing and communicating. In traditional WSN applications, the workloads are very simple and wireless communication is usually the most energy intensive process [14]. Specifically, a single bit transmission requires 1000 times the energy cost of a 32-bit computation in a classical architecture [15]. Thus, most of the previous researchers mainly focus on reducing the communication cost, at the expense of increasing the computation cost. For example, energy efficient clustering and routing approaches have been proposed to conserve communication energy by reducing the transmission distance and balancing the transmission loads within the clusters [16, 17, 18]. Alternatively, there are numerous compression-based techniques that either focus on reducing the volumes of the transmitting packets [19, 20, 21] or aim to decrease the transmission rate to achieve the energy efficiency [22, 23, 24, 25]. Also, a large number of sleep/wakeup schemes have been studied to reduce the energy spent on idle states of the radio component [26, 27].

However, as more complex applications have been implemented in WSNs during the past decade, such as Internet of Things or in-network processing, the computation energy consumption is comparable with or even larger than the communication cost [28, 29]. Executing those computationally intensive applications may make the sensor nodes die soon if the workloads of the tasks are not fairly distributed. In order to achieve energy efficiency and to extend the network lifetime, it is necessary to consider and balance properly the workloads of not only the communication tasks but also the sensing and processing tasks among the sensor nodes. Energy-aware task allocation, as one efficient solution to solve the energy balance problems, is starting to attract the attention of the WSN research community. Although task allocation approaches have been deeply studied in multiprocessor systems, grid computing and system on chip (SoC), the limited battery power and computing capability of sensor nodes, as well as the wireless communication in WSNs, make the problems difficult. The design of task allocation approaches for WSNs is an ongoing research.

1.2 Main Contribution

The key objective of the task allocation is to prolong the network lifetime. It does not mean just focusing on reducing the energy consumption of a single node, but rather the energy cost of all of the sensor nodes in the network should be considered and balanced. If an improper task allocation scheme is used, some sensor nodes in the network would quickly run out of energy because of the overload, even if other sensor nodes still have plenty of residual energy. As a result, the network cannot provide enough information for adequate time duration. To address this problem, this work proposes suitable energy aware task allocation algorithms to extend the network lifetime. The main contribution includes:

- This work provides an application level taxonomy and an indepth analysis of the task allocation approaches used in WSN. Moreover, this work illustrates a systematic formulation of the energy consumptions of the sensor nodes in WSNs by taking the processing, communicating and sleeping energy cost into account. Besides, different kinds of directed acyclic graphs (DAGs) are used to model the various WSN applications. The results of taxonomy of task allocation approaches and the detailed communication energy model have been reported in [30, 31], respectively.

- This work proposes a centralized static task allocation (CSTA) algorithm, a centralized dynamic task allocation (CDTA) algorithm and a distributed optimal on-line task allocation (DOOTA) algorithm for cluster based WSNs. Firstly, the task allocation problem for cluster based WSNs is formulated by partitioning each of the modeled DAG graphs into two subgraphs: one for the slave node and the other is executed by the master node. Then, CSTA algorithm is proposed. It formulates the workload distribution problem as a binary integer linear programming (BILP) problem by introducing binary vector variables to represent the partition cuts. This algorithm is executed by the gateway and provides the static partition solutions which enables each slave node to apply one fixed time invariant partition cut to balance the workload distribution of the tasks. Next, CSTA is extended to CDTA by using multiple partition cuts with different weights to achieve more balanced workload distribution among all of the slave and master nodes. By using the average energy cost of the slave and master nodes at each DAG execution round, CDTA formulates the dynamic task allocation problem as a linear programming (LP) problem. Furthermore, to overcome the drawbacks of the centralized algorithms, DOOTA is further proposed. It firstly presents an indepth analysis to prove that the optimal partition solutions consist of at most two partition cuts. Based on the extracted important partition cuts, DOOTA enables the slave and master nodes to negotiate on-line to calculate the optimal partition solutions. The proposed task allocation algorithms are important for WSNs, but also they can be used in other domains, e.g., mobile edge/fog computing. The results of CSTA, CDTA and DOOTA have been reported in [31, 32, 33], respectively.

- This work further extends the proposed task allocation algorithms for different task scenarios and network structures. Firstly, the scenario of condition triggered tasks is considered. Given a specific condition triggered application, it is modeled by a DAG graph with conditional branches. According to the conditional branches, i.e., the conditions of the tasks, the conditional DAG graph is classified into multiple stationary DAG graphs without conditional branches. The task allocation problem for condition triggered tasks is then modeled by dividing each of the stationary DAG graphs into two subgraphs simultaneously. Based on this model, this work extends CSTA and CDTA algorithms and proposes a static and a dynamic condition triggered task allocation algorithms (SCTTA and DCTTA), respectively. Next, this work proposes a static and a dynamic joint task allocation (SJTA and DJTA) algorithms for the applications with joint local and global tasks. The modeling of local task allocation is the same as CSTA and CDTA, while the global task allocation is modeled as dividing the global DAG graph into multiple subgraphs mapping to all of the slave and master nodes. This joint task allocation problem is formulated as a BILP and a LP problems by SJTA and DJTA algorithms, respectively. In addition to different task scenarios, this work also presents a dynamic task allocation (DTA-mhop) algorithm for multi-hop mesh networks. The multi-hop network task allocation is modeled by dividing the DAG graph of each sensor node into multiple subgraphs mapping to itself, the routing nodes and the sink node. DTA-mhop uses the summation of the task allocation solutions for each sensor node and formulates the problem of maximizing the network lifetime for multi-hop WSNs as a LP problem. The result based on the study of joint task allocation has been reported in [34], which is currently under review of an international conference, while the other two studies are in preparation for submission.

- This work validates the proposed task allocation algorithms by both extensive simulation results and physical implementations, using real WSN applications. Firstly, the proposed algorithms are evaluated in the simulated general WSNs. Then, a specific WSN hardware, OpenMote, is used to implement the proposed algorithms and evaluates their performance in experiments. The energy consumptions of the WSN nodes for applying the task allocation solutions, obtained by the proposed algorithms, are measured by visualizing the current profile on an oscilloscope.

1.3 Publications

The related publications of this work include [30, 33, 24, 31, 34, 32, 21, 25, 23], as shown below:

Book Chapters

- Wanli Yu, Yanqiu Huang, and Alberto Garcia-Ortiz. Energy Aware Task Allocation in Wireless Sensor Networks. In Habib M. Ammari, *The Philosophy of Mission-Oriented Wireless Sensor Networks*, in print by Springer, 2018.

Journal Articles

- Wanli Yu, Yanqiu Huang, and Alberto Garcia-Ortiz. Distributed Optimal On-line Task Allocation Algorithm for Wireless Sensor Networks. *IEEE Sensors Journal*, 18(1):446–458, Jan 2018.
- Yanqiu Huang, Wanli Yu, Christof Osewold, and Alberto Garcia-Ortiz. Analysis of PKF: A communication cost reduction scheme for wireless sensor networks. *IEEE Transactions on Wireless Communications*, 15(2):843–856, Feb 2016.
- Yanqiu Huang, Wanli Yu, and Alberto Garcia-ortiz. Accurate energy-aware workload distribution for wireless sensor networks using a detailed communication energy cost model. *Journal of Low Power Electronics*, 10(2):183–193(11), June 2014.
- Yanqiu Huang, Wanli Yu, and Alberto Garcia-Ortiz. EPKF: Analysis of optimal reconstruction methods based on incomplete information from sensor nodes. *Under review of IEEE Sensors Journal*.

Conference Proceedings

- Wanli Yu, Yanqiu Huang, and Alberto Garcia-Ortiz. Joint Task Allocation Approaches for Hierarchical Wireless Sensor Networks. *The 7th International Conference on Modern Circuits and Systems Technologies (MOCAST): Electronics & Communications*, pages 1–4, Thessaloniki, Greece, 2018.
- Wanli Yu, Yanqiu Huang, and Alberto Garcia-Ortiz. Modeling optimal dynamic scheduling for energy-aware workload distribution in wireless sensor networks. In *Proceedings of the 2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 116–118, Washington DC, USA, May 2016.

- Yanqiu Huang, Wanli Yu, and Alberto Garcia-Ortiz. PKF-ST: A communication cost reduction scheme using spatial and temporal correlation for wireless sensor networks. In *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks (EWSN)*, pages 47–52, Graz, Austria, Feb 2016.
- Wanli Yu, Yanqiu Huang, and Alberto Garcia-Ortiz. An altruistic compression-scheduling scheme for cluster-based wireless sensor networks. In *Proceedings of the 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 73–81, Seattle, USA, June 2015.
- Yanqiu Huang, Wanli Yu, and Alberto Garcia-Ortiz. PKF: A communication cost reduction schema based on kalman filter and data prediction for wireless sensor networks. In *Proceedings of the 26th IEEE Inernational system-on-chip conference*, pages 73–78. CAS, Sep 2013.

1.4 Dissertation Outline

This dissertation is organized in the classical form of three main parts: an introduction, where the state of the art and related background are stated; a central core, where the proposed task allocation algorithms are presented; and a final part with the validations of the approaches and the conclusion. More precisely:

I. **Introduction**: Chapter 2 and Chapter 3 introduce the state of the art and the background.

   This dissertation starts with an introduction of various WSN network structures and applications, and is followed by an indepth analysis of the state-of-the-art task allocation approaches in WSNs in Chapter 2. In the next chapter, a systematic formulation of the energy consumptions of the WSN nodes is firstly illustrated based on a detailed communication energy cost model. Then, Chapter 3 further presents the modelings of various WSN applications. These energy formulations and application models are the foundations for the proposed approaches in the following chapters.

II. **Core:** the proposed approaches are presented in Chapter 4 and Chapter 5.

   Chapter 4 firstly proposes the CSTA task allocation algorithm, which provides the time invariant task allocation solutions for cluster based WSNs to extend the network lifetime. Moreover, the CDTA algorithm is further proposed to provide dynamic task allocation solutions, thereby achieving longer network lifetime extension. Since both CSTA and CDTA are computationally intensive centralized algorithms, the very lightweight DOOTA is designed to improve the efficiency of on-line execution of the task allocation algorithm. Chapter 5 further extends the proposed algorithms for different task scenarios and network structures. Targeting the applications with conditional tasks, the SCTTA and DCTTA algorithms are developed; for applications that consist of joint local and global tasks, the SJTA and DJTA task allocation algorithms are proposed; and the DTA-mhop algorithm is further presented for the multi-hop network task allocation problem.

III. **Conclusions**: the validations of the approaches and the final conclusions are described in the last two chapters.

The performance of the proposed task allocation algorithms are firstly estimated based on simulations using real WSN applications. Moreover, one of the most popular WSN hardwares, OpenMote, is used to implement the proposed algorithms to measure the real improvement of the network lifetime. Finally, the last chapter concludes this work and points out the future research directions.

2 Background and Related Works

2.1 Introduction **9**
2.2 Wireless Sensor Networks **10**
2.2.1 Typical WSN Topology Structures 11
2.2.2 WSN Applications 13
2.3 Related Works for Task Allocation in WSNs **14**
2.4 Summary **18**

2.1 Introduction

WSNs have been applied to a wide variety of applications with vastly varying requirements and characteristics. Task allocation is performed to extend the network lifetime by taking the network topology structure, battery power and node on-board processing abilities into account. In WSNs, the deployed sensor nodes are connected by either single wireless hop or multiple wireless hops to collaboratively complete the tasks of the given applications. Due to the limited energy resources of the sensor nodes and the long time requirements of WSN applications, how to properly assign the tasks for each sensor nodes to extend the overall network lifetime becomes very crucial.

This chapter firstly introduces an overview of the network topology structures of WSNs and the applications in Section 2.2. Two typical types of network structures, hierarchical cluster based WSNs and multi-hop mesh WSNs, are presented. The applications are classified according to the types of the tasks. Then, Section 2.3 presents the related task allocation algorithms used in WSNs. These approaches are critically analyzed based on the optimality of the solutions and the types of tasks. The gaps in previous researches are outlined, which motivates the design of the approaches in Chapters 4 and 5. The last section concludes the work of this chapter.

2.2 Wireless Sensor Networks

The recent advances of micro-electro-mechanical-systems (MEMS), digital electronics and wireless communication techniques have enabled the development of low-cost, low-power and multifunctional sensor nodes. These sensor nodes are mainly made up of a sensing unit, a processing unit, a communication unit and a power unit [35]. Fig. 2.1 illustrates the components of a general WSN node. The sensing unit integrates one or more sensors and interfaces the physical world with the digital world based on the analog-to-digital converters (ADCs). The processing unit, consists of processor and memory, is capable of on-board processing and brings together all the other units and some additional peripherals. The microcontroller (MCU) is one of the most popular processors used by WSN nodes[1]. The communication unit is in charge of transmitting and receiving the data by using the radio wave. The typical power source of a sensor node is battery energy.

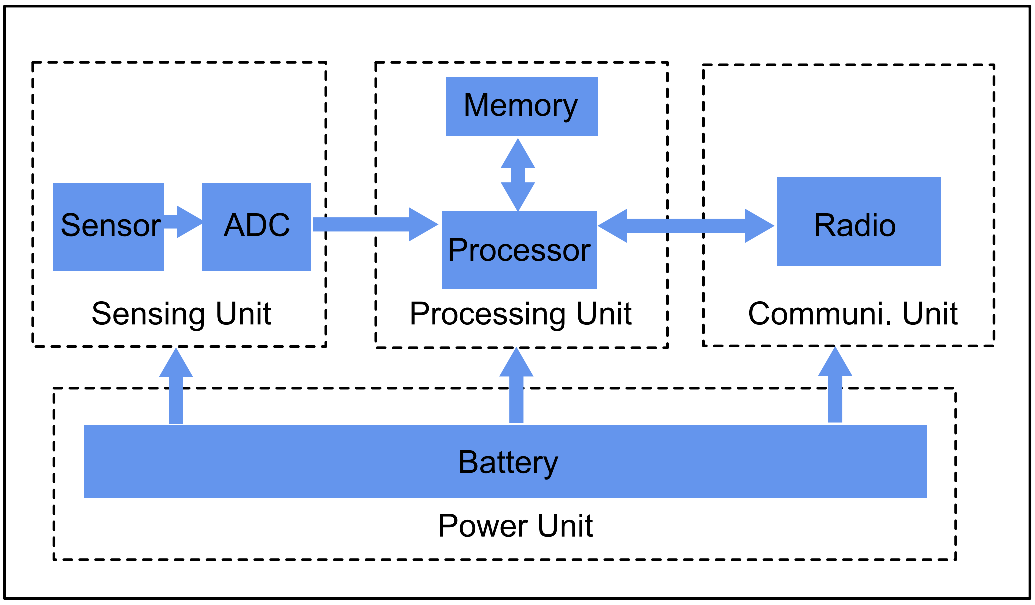


Figure 2.1: Components of a general WSN node.

In WSNs, the sensor nodes are connected by either single hop or multi-hop wireless communications based on different network topology structures to collaboratively complete various applications. This section introduces the basic network structures of WSNs and briefly overviews the related applications.

[1]There are also other options to built the processing unit, such as using digital signal processor (DSP), application specific integrated circuit (ASIC) or field programmable gate array (FPGA).

2.2.1 Typical WSN Topology Structures

For concreteness, this section introduces two typical network structures: multi-hop mesh and hierarchical cluster networks[2].

- **Multi-hop Mesh WSNs**

  One of the most popular network structures used in WSNs is the multi-hop mesh network structure. In multi-hop mesh WSNs, all the sensor nodes are considered equal with respect to their roles and functionalities. In addition to the abilities of sensing, processing and transmitting, each sensor node also can operate as a routing node. The observation of each sensor node is propagated by multiple wireless hops from the first routing node to the next one until the observation reaches the destination (sink node). The selection of the transmission path depends on the used routing protocols, such as the minimum hop routing protocol which has been widely used in numbers of studies [36, 37, 38, 39]. Since the wireless transmission range is limited, only the neighbor nodes within the transmission range of the sender are able to act as the routing nodes. Besides forwarding the data, the routing nodes can also share the processing tasks for the sender. Fig. 2.2 shows an example of the multi-hop mesh WSNs. The advantages of multi-hop mesh network structure can be basically concluded as: inexpensive network structure, easiness of implementation and broadband data support [40].

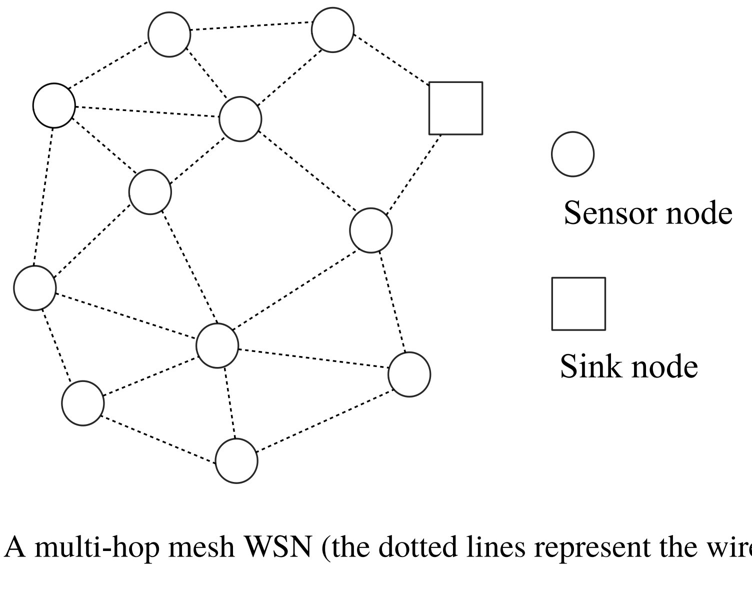


Figure 2.2: A multi-hop mesh WSN (the dotted lines represent the wireless hops).

[2]Note that, the network structures of WSNs are not limited to multi-hop mesh and hierarchical cluster, there are also other types, such as chain based or location based network structures, etc.

- **Hierarchical Cluster WSNs**

Cluster based network structure has been widely used in WSNs. In hierarchical cluster WSNs, all the sensor nodes are grouped into a number of small sized clusters according to various clustering approaches such as the famous LEACH protocol [41]. In each cluster, there are usually one coordinator termed as cluster head (also named as master node) and a number of leaf nodes (also referred to as slave nodes). Clustering typically results in a two-tier hierarchy: the master nodes and the slave nodes form the higher and lower tiers, respectively. Fig. 2.3 illustrates the data flow in a cluster based WSN. Each slave node is in charge of sensing and then either directly transmitting the raw data to its master node or pre-processing the data before the transmission. The master node is in charge of receiving data from its slave nodes and executing further processing; after that, it forwards the processed data to the sink node by either direct transmission or multiple hops among the master nodes. The workloads of the given applications are completed through the collaboration of the slave nodes and the master nodes. Typically, each cluster can work independently, many studies mainly focus on the activities within the cluster [21, 31, 29, 32, 33, 42]. Consequently, clustering is particularly useful for applications that require scalability to hundreds or thousands of nodes.

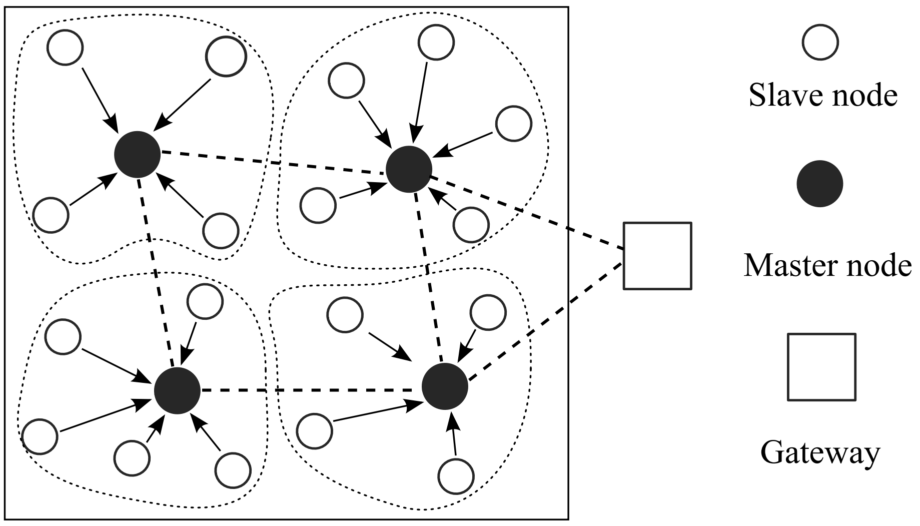


Figure 2.3: A hierarchical cluster-based WSN (dotted lines represent the wireless hops among the master nodes).

2.2.2 WSN Applications

As previously described, WSNs have been applied to a wide variety of applications in various domains. A number of studies have been devoted to categorize the WSN applications according to its application domains [43, 44]; it is almost impossible to list all of the possible application areas of WSNs. Instead, this section focuses on analyzing the types of the tasks in the applications.

In WSNs, the related applications can be basically grouped into two categories: static and dynamic applications, according to whether the tasks of the applications are time invariant or not.

- **Static Applications**

  Static applications are made up of a fixed number of tasks. Since WSN nodes are integrated with one or more sensors and capable to do the on-board processing, the application can be executed in each node or in collaboration with others according to the complexities of the tasks. Once a static application is executed, the number of the tasks and the overall workload of the application are time invariant. This kind of applications is one of the most frequently used WSN applications in our daily lives. For example, the WSN-based pipeline monitoring applications, such as PipeNet [45], EARNPIPE [46], have been developed to monitor hydraulic and water quality by measuring water pressure, pH, etc. More precisely, in [45], the pressure data is collected by each deployed sensor node every 5 minutes for a period of 5 seconds with the sampling rate of 100 Hz. Another example of static WSN application is weather monitoring application [47, 48, 49]. In order to analyze and predict the future weather, the WSN nodes are required to periodically measure the temperature or the humidity of the interested places year by year. Besides, static WSN applications have been also used in underground coal mine environment, e.g., in [50], the specially designed WSN nodes are carried along with the coal miners to measure their heart beats and the nearby methane concentrations all the time.

- **Dynamic Applications**

  The applications, composed of different tasks as time evolves, are categorized as dynamic applications. A key characteristic of dynamic applications is that they can include some condition requirements. According to whether the conditions are satisfied or not, the sensor nodes are required to execute different tasks. The condition triggered dynamic applications are starting to attract a lot of attention in the past few years. For instance, the transmission rate reduction approaches, such as [23, 51, 52, 53], enable the WSN nodes to execute different tasks according to the defined conditions. Specifically, in [23, 51], the sensor nodes process the measured data and compare with the result of one predictor. When the error between the measured and predicted data exceeds a predefined threshold, the nodes transmit date. Therefore, the nodes need to execute different tasks as the time-varying noise changes.

2.3 Related Works for Task Allocation in WSNs

In a resource constrained WSN, effective task allocation is very important for balancing the energy consumptions among the nodes and extending the network lifetime. Although numerous task allocation approaches have been studied in previous wired networks [54, 55], they cannot be directly applied to WSNs due to the limited battery energy and the special wireless communication mechanism. The design of task allocation approaches for WSNs is an ongoing research and attracting significant attentions.

In [56], the authors have studied the energy balanced task allocation problem to extend the network lifetime. They consider an epoch-based application scenario which is made up of a set of communication tasks, and formulate the problem of maximizing the network lifetime by assigning suitable tasks for the sensor nodes as an integer linear programming (ILP) problem. However, this work only focuses on distributing the communication tasks and has not taken the tasks generated by the applications into account. Furthermore, it aims at the homogeneous networks, which are not common in realistic scenarios. The authors in [57] propose a tree-structured linear approximation scheme by considering the heterogeneous properties of sensor nodes. It enables the sensor node to transmit based on the calculated best-fit piecewise partition of the communication tasks. Still, this work does not consider the effect of computation energy cost.

In addition to distribute the communication tasks, a group of studies focus on minimizing the execution time of the WSN sensing tasks to maximize the network lifetime [58, 59, 60, 61], because quick response of a WSN could make for energy saving. They mainly aim at eliminating transmission collisions and idle gaps between two successive data transmissions for cluster-based WSNs. In the networks, the slave nodes firstly do the sensing tasks then transmit the collected data to the master node; the master node works as a data fusion node and transmits the processed data to the sink node. The task allocation and scheduling include two stages: intra-cluster task scheduling and inter-cluster task scheduling. The procedure for the sensing task allocation can be briefly summarized as follows: firstly, the sink node distributes the tasks to each cluster using inter-cluster task scheduling; then, the intra-cluster task scheduling arranges the subtasks to each leaf node in the cluster. Nonetheless, these approaches only focus on allocating the sensing tasks, which is not sufficient in current WSNs.

As mentioned in Section 1.1, WSNs have been applied to more and more complex applications during the last decade, e.g., IoT or in-network processing. The computation energy cost becomes comparable with the communication cost. In order to effectively extend the network lifetime, a number of studies develop task allocation algorithms by considering the computation and communication energy cost simultaneously. Some approaches are able to provide the optimal solutions, while others only can supply the suboptimal solutions. Normally, obtaining the optimal solutions increases the computational cost dramatically. The task allocation problem in WSNs has been proved to be NP-hard [29]. Thus, many works focus on the heuristic algorithms to obtain the suboptimal task allocation solutions with lower complexity. The category of the heuristic task allocation algorithms can be classified into two main groups: the *traditional heuristic* and the *bio-inspired heuristic* algorithms.

- **Traditional heuristic task allocation algorithms**

  Traditional heuristic algorithms mainly reduce the complexity by seeking the local optimal solutions. The authors in [29] adopt the ideas from the Kernighan-Lin (K-L) [62] and Fiduccia-Mattheyses (F-M) [63] algorithms and develop a heuristic task allocation approach for cluster based WSNs. The task allocation problem is defined as an energy-driven partitioning (EDP) problem, while considering the energy costs associated with computation and communication. The maximum energy cost of the slave and master nodes is selected as the cost function. Based on an initial partition cut, the proposed algorithm incrementally exchanges the vertexes (tasks) across the partition cut. Every time, the vertex with the biggest positive gain (the difference of cost function between before and after moving the vertex to the other side) will be moved and then blocked. This algorithm ends until there is no free vertex with positive gain. Through this mark and block mechanism, the complexity of task allocation is significantly reduced. Similarly, in [64], an energy balancing task scheduling and allocation heuristic for extending the network lifetime, termed as EBSEL, is proposed. It tries to minimize the communication cost without sacrificing parallelism, and employs a thresholding technique to avoid the early death of the nodes. However, these approaches are restricted to homogeneous networks and only provide static task allocation solution.

  In the last few years, a number of new heuristic task allocation algorithms have been proposed based on game theory [65, 66, 67, 68]. In such works, the task allocation is modeled as a market-based architecture which mainly includes: *mission manager*, *auctioneer*,*bidders* and *service chart* as depicted in Fig. 2.4. Given a specific application, the *mission manager* firstly models the application and then delivers the tasks to the *auctioneer* according to their priorities. When the *auctioneer* receives the task, it transmits a *task message*, which is made up of task size and task deadline, to the *bidders*. Typically, the sink node plays the role of the *auctioneer*. The *bidders* are the sensor nodes. After each *bidder* receives one task message from the *auctioneer*, they calculate their cost price for accomplishing the task based on the task processing energy consumption, the involved communication cost, the task deadline and their own residual battery energy. Then, the *bidders* send their bids to the *auctioneer*. According to the cost prices, the *auctioneer* makes its decision to choose the winner and assigns this task to it. The cost price of the winning *bidder* is recorded in the *service chart*, which will be accumulated to the winning *bidder* when it calculates the cost price of the next task message. Although these approaches are able to balance the energy consumption of the network, the efficiency of them in large networks still needs to be validated. Moreover, getting stuck in the local optimum is a common drawback for most of the traditional task allocation algorithms.

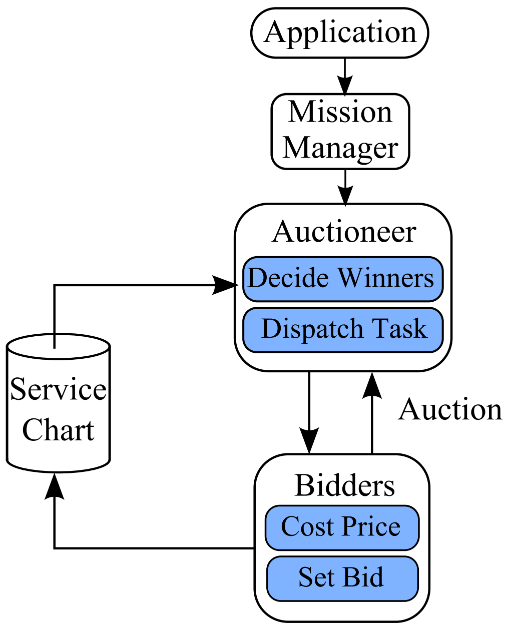


Figure 2.4: Market-based architecture for the task allocation in WSNs (Modified from [65, 66]).

- **Bio-inspired heuristic task allocation algorithms**

Compared with the traditional heuristic algorithms which usually get stuck in local optimum, the bio-inspired heuristic algorithms adopting the bionic intelligence can obtain the global optimal solutions with high probability. Among the current studies, genetic algorithm (GA) [69, 70, 71, 72] and particle swarm optimization (PSO) [73, 74, 75, 76] are widely employed.

GA is a heuristic search and optimization technique that imitates the biological process of natural evolution. It is used to search the "fittest" solutions for a given maximization or minimization problem. Almost all genetic algorithms have at least the following common components: genetic representations of the solution domains (generations of chromosomes), a fitness function for optimization, selection according to fitness, crossover to produce new generation and random mutation in new generation [77]. The authors in [71] propose an energy balance DAG task scheduling algorithm based on GA to find the better task allocation solution in the whole solution space. In [69, 70], GAs are adopted to address the task allocation problems in multi-hop wireless sensor networks. A slight difference from general GA, an inheritance progress is used before the selection, which enables the good genes inherited to the next generation. Since the natural evolution tells us that there is a higher probability of generating better offspring if both the parents have good genes, GA based task allocation algorithms have high probability to find the global optimum.

PSO is another popular heuristic optimization algorithm which is inspired by social behavior and movement dynamics of some animals such as the flocking behavior of birds and the schooling behavior of fish. The general idea of PSO is that a population of particles (candidate solutions) repeatedly move to the next positions (modified solutions), according to the guides of each one's personal best solution, the global best solution among the population and the current movements of the particles, until the requirements are satisfied. Like GA, a fitness function is also needed in PSO for the optimization to update the better solutions. The task allocation problem in WSNs can be described as a 0-1 decision problem where each boolean parameter indicates whether the node is selected for a task. Therefore, binary PSO (BPSO) and its modified versions have been commonly employed in [73, 74, 75, 76], by representing each individual particle as a binary string. Like GA algorithm, PSO based task allocation algorithms also cannot guarantee the global optimal solutions.

In contrast to the traditional heuristic task allocation algorithms, the bio-inspired algorithms perform better on the network lifetime improvement and energy balance among the sensor nodes. Nevertheless, neither GA or PSO guarantees the global optimum task allocation solutions. Besides, they need a large quantity of iterations of the generation to converge, which could consume longer time. Moreover, most of the relevant works only provide static task allocation solutions.

According to the above analysis of the related task allocation approaches for WSNs, the goal of this work is to provide optimal task allocation solutions to maximize the network lifetime by considering both the computation and communication energy cost. As mentioned in Section 2.2.1, each cluster can be optimized independently. The size of a typical cluster, i.e., the number of the leaf nodes in the cluster, is relatively small, so that the complex optimal task allocation algorithms are affordable in cluster based WSNs. Therefore, Chapter 4 presents a centralized static task allocation (CSTA) algorithm and a centralized dynamic task allocation (CDTA) algorithm for cluster based WSNs based on binary integer linear programming (BILP) and linear programming (LP), respectively. Moreover, a lightweight distributed optimal on-line task allocation (DOOTA) algorithm is further presented in Chapter 4 to overcome the complexity drawbacks of the centralized algorithms. In order to cover different network structures and task scenarios, the extensions of CSTA and CDTA are further presented in Chapter 5.

2.4 Summary

This chapter has briefly introduced the basic characteristics of WSNs and reviewed the related works for task allocation in WSNs.

Firstly, two typical network topology structures, multi-hop mesh and hierarchical cluster network structures, have been presented. It is followed by a novel taxonomy of the WSN applications, which groups the applications into two categories: static and dynamic applications.

Then, the related works for task allocation in WSNs have been reviewed. Typically, a WSN application consists of computation and communication tasks. Only focusing on one of them is not sufficient, since computation energy cost is becoming comparable with communication cost in current applications. Therefore, a number of task allocation algorithms have been developed by considering both the computation and communication cost. Because of the complexity for obtaining optimal solutions, designing the heuristic algorithms is one of the main focuses by now. Traditional heuristic task allocation algorithms have been firstly analyzed. Some of them are either restricted to homogeneous networks or static task allocation solutions. Further more, they always suffer from getting stuck in the local optimum. In contrast to traditional heuristics, bio-inspired heuristics have been conducted, i.e., GA and PSO, by researchers for obtaining the global optimum. However, these bio-inspired heuristics do not guarantee the global optimal solutions, and also they need a large quantity of iterations, which is very time consuming.

The limitations in state-of-the-art works motivate this work to design optimal task allocation algorithms to maximize the network lifetime by taking both the computation and communication energy cost into account. Before presenting the proposed algorithms, the models of the energy cost of sensor nodes and the applications are firstly given in Chapter 3. Then, the detailed descriptions the proposed algorithms are presented in Chapters 4 and 5, respectively.

3 System Models Used for Task Allocation Problem

3.1 Introduction **21**

3.2 Modeling WSN Applications **22**

3.2.1 Graphs 22

3.2.2 WSN Application Models 23

3.3 Modeling the Energy Consumptions of WSN nodes **24**

3.3.1 Cost Functions 25

3.3.2 A Detailed Communication Energy Cost Model for WSN Nodes . . . 26

3.3.3 Validation of the Proposed Communication Energy Cost Model . . . 29

3.4 Evaluation Metrics for Task Allocation Algorithms **32**

3.5 Summary **35**

3.1 Introduction

The objective of the proposed task allocation algorithms is to extend the network lifetime by suitably assigning the tasks of applications. To achieve this objective, a key requirement is the accurate system models, which mainly involves the applications, energy consumptions of nodes and the evaluation metrics of task allocation algorithms.

This chapter firstly models the static and dynamic applications by standard directed acyclic graph (DAG) and DAG graph with conditional branches, respectively, in Section 3.2. Then, Section 3.3 describes the energy cost functions of WSN nodes including the energy cost of processing, communicating and sleeping. In order to accurately formulate the energy cost of the wireless communication activities, this section proposes a detailed communication cost model. Afterwards, Section 3.4 categorizes the evaluation metrics for task allocation algorithms. Finally, Section 3.5 summarizes and concludes this chapter.

3.2 Modeling WSN Applications

This section models the applications by DAG graphs. It firstly introduces the fundamental concepts of graphs in Section 3.2.1. Based on the basics of graphs, Section 3.2.2 presents the modeling of static and dynamic applications.

3.2.1 Graphs

Graphs are mathematical structures used to model pairwise relations between objectives. A graph is formed by vertexes and edges connecting the vertexes [78]. It consists of a pair of sets, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ is the set of vertexes and $\boldsymbol{E}$ is the set of edges formed by pairs of vertexes to describe their relationships. The edges in a graph can either have direction or not. A graph with directed edges is called directed graph, which will be used in this dissertation. Fig. 3.1 depicts one example of a directed graph with 5 vertexes and 5 edges.

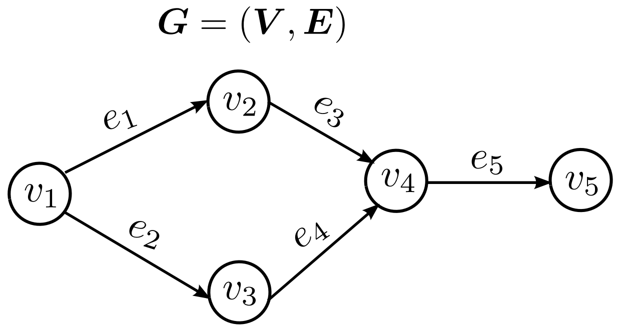


Figure 3.1: An example of directed acyclic graph with 5 vertexes and 5 edges.

Given a specific graph, it can be represented by the matrix formats, e.g., the adjacency matrix and the incidence matrix. The adjacency matrix of a directed graph is an $n \times n$ matrix $\boldsymbol{A}$, where n is the number of vertexes in the graph. Each element $\boldsymbol{A}(i, j)$ equals the number of edges that come out of vertex v_i and go into vertex v_j. Taking the directed graph in Fig. 3.1 for example, its adjacency matrix is:

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The incidence matrix of a directed graph is an $n \times m$ matrix $\boldsymbol{B}$, where n and m are the number of vertexes and edges in the graph, respectively. In $\boldsymbol{B}$, each element satisfies:

$$\boldsymbol{B}(v, e) = \begin{cases} 1, & \text{if edge } e \text{ comes out from vertex } v \\ -1, & \text{if edge } e \text{ enters into vertex } v \\ 0, & \text{otherwise} \end{cases}$$

The corresponding incidence matrix of the graph in Fig. 3.1 is expressed as follows.

$$\boldsymbol{B} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

3.2.2 WSN Application Models

Typically, WSN applications consist of a set of processing tasks and a paired set of communication tasks. Since the processing tasks are usually with orders, Directed Acyclic Graphs (DAGs) have been widely used to model the WSN applications, such as in [31, 69, 73, 32]. In a DAG graph, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, each vertex $v \in \boldsymbol{V}$ represents a task of the application and connects with others by directed edges. The computational workload of the vertex v is typically represented by the total number of CPU clock cycles of the WSN node for executing the task. It is denoted as $w(v)$ in this work. Each directed edge $e \in \boldsymbol{E}$ stands for the communication from its source vertex $src(e)$ to its sink vertex $snk(e)$. The weight of edge e is denoted as $l(e)$ that represents the amount of transmitted data in bits. Each task v consumes the data received from its predecessors, generates data and transmits to its successors. The predecessors always have the higher priorities, which means that the successor cannot be executed until it receive the data from all of its predecessors. For example, in Fig. 3.1, if task v_4 needs to be executed, it has to firstly receives the data from its predecessors v_2 and v_3. A task without any predecessor is called *source task*, which is typically assumed as the sensing task[1]. It does not have any predecessor and has the highest priority.

For a static application, it can be modeled by a standard DAG graph, e.g., the graph shown in Fig. 3.1. Towards the dynamic applications, the tasks that need to be executed may change according to different conditions as time goes. For example, the algorithm PKF proposed in [23] enables each slave node to transmit data until the calculated errors beyond the defined threshold. Therefore, an application with conditional tasks can be modeled by a DAG graph with conditional branches. Typically, the satisfaction probability of each condition can be measured by executing the application over a period. Based on the satisfaction probabilities of conditions, Fig. 3.2 illustrates one example of a DAG graph with two conditions. According to the requirements of the conditions, different vertexes and edges will be selected.

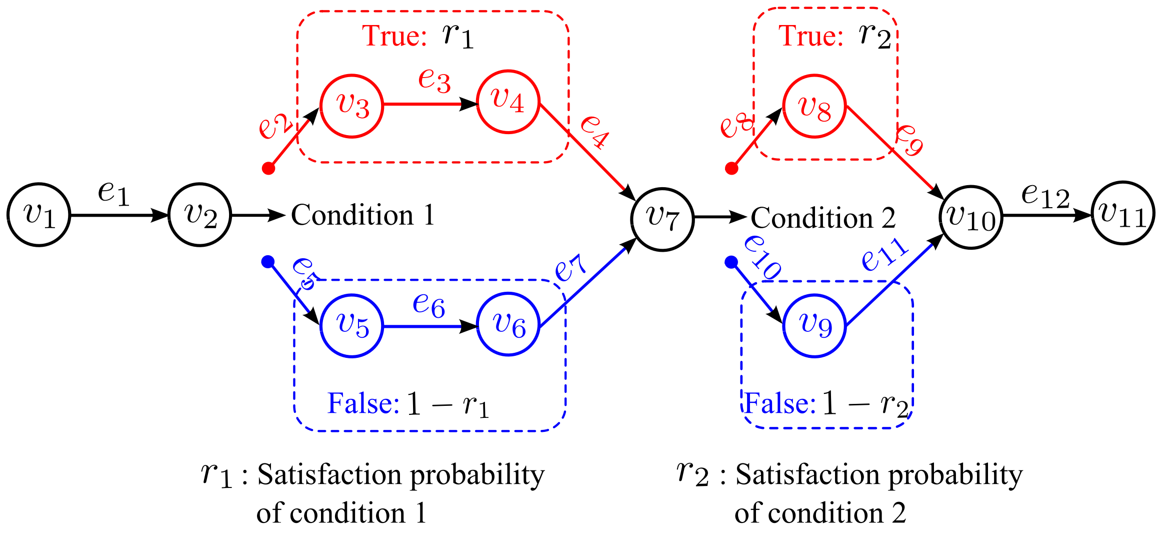


Figure 3.2: An example of a DAG graph with conditional branches (there are two conditions in the application, r_1 and r_2 are the satisfaction probabilities of condition 1 and condition 2, respectively).

Since the nodes are integrated with one or more sensors, each of them can be treated as individual source and can be used to accomplish different applications.

3.3 Modeling the Energy Consumptions of WSN nodes

The network lifetime depends on the energy consumption of each WSN node in the network. This section formulates the energy cost of the activities of the node.

[1]The number of the *source tasks* are not limited to one, it is determined by specific applications.

3.3.1 Cost Functions

In a WSN, each node mainly spends energy on processing the tasks, transmitting or receiving the data and sleeping. Note that the sensing tasks can be included in the processing tasks according to [29].

Processing energy cost:

The execution time of WSN node i for executing task v, $t_i(v)$, can be formulated as:

$$t_i(v) = \frac{w(v)}{f_i} \tag{3.1}$$

where $w(v)$ is the processing workload of task v, f_i is the processing speeds of node i (unit, Hz). The corresponding processing energy consumption of node i, E_{pi}, can be formulated as:

$$E_{pi}(v) = P_i \; t_i(v) \tag{3.2}$$

where P_i is the average processing power of node i (unit, $J/Sec.$).

Communication energy cost:

In order to accurately formulate the communication energy cost of WSN node, this work proposes a detailed communication energy cost model. According to the proposed model, the communication activities of a WSN node includes not only the data packets communication but also the overhead activities such as radio startup, channel access, etc. Consequently, the energy cost for transmitting and receiving L bits of data, E_{tx} and E_{rx}, can be expressed as:

$$E_{tx} = e_o + e_{tx}(d_i)\, L \tag{3.3}$$

$$E_{rx} = e_o + e_{rx}\, L \tag{3.4}$$

where e_o is the energy consumption of the overhead activities, d_i is the transmission distance of node i, $e_{tx}(d_i)$ and e_{rx} are the energy dissipated by transmitting and receiving one bit of data package, respectively.

The detailed description of the proposed communication cost model and its validation are presented in Sections 3.3.2 and 3.3.3, respectively.

Sleeping energy cost:

After executing the processing and transmitting/receiving options, the WSN node i will go to sleep mode to save energy. Correspondingly, the sleeping energy cost of node i, E_{slp_i}, is usually formulated as:

$$E_{slp\_i} = P_{slp\_i}\, t_{slp\_i} \tag{3.5}$$

where P_{slp_i} and t_{slp_i} are the average power consumption and time duration of node i in sleeping mode.

3.3.2 A Detailed Communication Energy Cost Model for WSN Nodes

This section presents the detailed information of the proposed communication energy cost model for WSN nodes based on previous literatures.

In the real communication process, a sensor node will firstly turn on the radio when it wants to send or receive data. Before transmitting the data packet, the node needs to access the wireless channel and possibly exchange some control packets according to some MAC protocols. After that, the actual transaction commences. Once the transaction accomplished, the radio is shut down. During this period, the node turns on its receiver prior to the actual reception because of the unawareness of the destination active state (it is the so called idle listening). It is possible to receive some packets that are not intended for it (namely overhearing). Due to the existence of collision, the packets may not be transmitted or received successfully, which causes retransmission and extra energy cost. Thus, the total communication energy consumption of a WSN node (from the radio startup to shutdown), E_{cmn}, can be described with the following terms:

$$\begin{aligned} E_{cmn} =& E_{startup} + E_{channel\_access} + E_{control} + E_{turnaround} \\ & + E_{idle\_listening} + E_{overhearing} + E_{collision} + E_{data\_packet} \end{aligned} \tag{3.6}$$

which corresponds to the energy cost of radio startup, channel accessing, control packets, turnaround, idle listening, overhearing, collision and data packets transmission, respectively

In order to calculate the energy cost of each part, the above-mentioned communication process, from radio startup to shutdown, is divided into seven different states as listed in Table 3.1.

By using the number of cycles, exchanging bits and the duration in each state (see Table 3.2

Table 3.1: Seven different states of a WSN node in the communication process.

State	**Definition**
ST	Node turns on the ratio
CC	Node tries to access the wireless channel
IL	Node turns on its receiver prior to the real-time receiving
OH	Node receives data packets that are not intended to it
RX	Node receives the data packet
TX	Node transmits the data packet
TA	Node switches its state between RX and TX

for details), Eq. (3.6) can be re-expressed as:

$$E_{cmn} = N_{st}e_{st} + P_{cc}t_{cc} + (e_{rx}L_{cr} + e_{tx}L_{ct}) + N_{ta}e_{ta} + P_{il}t_{il} + e_{oh}L_{oh} + E_{cl} + e_{tx/rx}L_d \tag{3.7}$$

The collision energy cost E_{cl} is affected by several parameters, e.g., the collision probability r_{cl}, the channel accessing failure probability r_{cc}, etc. It is convenient to calculate the average communication energy cost for each packet by introducing the average transmission times $N(r_{cc}, r_{cl})$ as proposed in [79, 80]. According to $N(r_{cc}, r_{cl})$, Eq. (3.7) is further formulated as:

$$\begin{aligned} E_{cmn} =& N_{st}e_{st} + N(r_{cc}, r_{cl})\Big(P_{cc}t_{cc}(r_{cc}) + (e_{rx}L_{cr} + e_{tx}L_{ct}) + N_{ta}e_{ta} \\ &+ P_{il}t_{il}(r_{cc}) + e_{oh}L_{oh}(r_{cc}) + e_{tx/rx}L_d\Big) \end{aligned} \tag{3.8}$$

It is important to note the different nature of the parameters used by the model. They are either **constant** (independent of the transmission pattern) or **variable** (associated with the communication scenario). Furthermore, they are determined by the hardware (**HW**), the **MAC** protocol, the application layer (**APP**) or their combinations as listed in Table 3.2.

Constant parameters include: the number of control packets, the radio startup and turnaround times (specified in the MAC protocol) as well as the energy consumption of the corresponding states. Their values are determined by the HW platform. Typically, the energy cost of receiving and transmitting one bit of data packet, e_{rx} and e_{tx}, can be expressed by Eqs. (3.9) and (3.10) [81].

$$e_{rx} = P_{T0}t_{rx} \tag{3.9}$$

$$e_{tx}(d_i) = \Big(P_{T0} + \frac{10^{(L(d)+P_{rin})/10}}{\eta}\Big)t_{tx} \tag{3.10}$$

Table 3.2: Parameters in the communication energy cost model of WSN nodes.

constant	**HW**	e_{st}	The energy cost of radio startup
		P_{cc}	The power cost of accessing channel
		$e_{tx}(d)$	The energy cost of transmitting one bit of data (as a function of the distance d)
		e_{rx}	The energy cost of receiving one bit of data
		e_{ta}	The energy cost of turnaround
		P_{il}	Idle listening power cost
		e_{oh}	The energy cost of overhearing one bit of data
	MAC	L_{cr}	Receiving bits in control packets
		L_{ct}	Transmitting bits in control packets
		N_{ta}	The number of turnaround times
		N_{st}	The number of radio startup times
variable	**APP**	L_d	Transmitting bits in data packets
		T_{scd}	The duration of a scheduling period
	APP&MAC	$N(r_{cc}, r_{cl})$	The average number of transmission times for each packet (as a function of r_{cc},and r_{cl})
		r_{cc}	Clear channel assessment (CCA) failure probability
		r_{cl}	Collision probability
		$t_{cc}(r_{cc})$	Mean access channel time
		$t_{il}(r_{cc})$	Mean idle listening time
		$L_{oh}(r_{cc})$	The mean number of overhearing bits

where P_{T0} is the energy consumed by the electronic circuits of the transceiver for receiving or transmitting 1 bit of data, P_{rin} is the receiver sensitivity, η is the drain efficiency, t_{rx} and t_{tx} are the time of receiving and transmitting 1 bit of data, respectively, and $L(d_i) = 20\log_{10}(F) + 10\alpha\log_{10}(d) - 27.55$ in which F is the RF frequency and α is the path loss exponent.

Variable parameters determined by application layer are the number of transmitting bits L_d and the scheduling period T_{scd}. In addition, all the parameters related to the channel condition are variable as well. They depend on which MAC protocol the sensor node uses, the number of contending nodes and the communication frequency stated by the application layer.

The variable parameters determined by TDMA based protocols can be simply obtained as presented in [31]. This communication energy cost model mainly focuses on CSMA/CA based MAC protocols to formulate the variable parameters. The successful channel access probability of a node is $1 - r_{cc}^{MaxNB+1}$, where r_{cc} is the probability of channel access failure and $MaxNB$ is the maximum channel access repetitions. Combining the collision probability, r_{cl}, the average transmission times for each data packet, $N(r_{cc}, r_{cl})$, can be approximated as:

$$N(r_{cc}, r_{cl}) = \sum_{i=0}^{Maxretry} (1 - r_{cc}^{MaxNB+1})^{i+1} r_{cl}^{i} \tag{3.11}$$

where $Maxretry$ is the maximum retransmission times specified by the MAC protocols. For the detailed calculation of r_{cc} and r_{cl} based on the number of contending nodes, please refer to [80]. In one data packet transmission attempt, a sensor node would send in average $\sum_{i=0}^{MaxNB} r_{cc}^{i}$ CSMA trials. In each trial, the node waits in average $\frac{1}{2}(2^{min(MinBE+i,MaxBE)} - 1)t_{bk}$ back-off duration and then executes the clear channel assessment (CCA) procedure in the following t_{cca} period. Thus, the average channel access time for each transmission is:

$$t_{cc}(r_{cc}) = \sum_{i=0}^{MaxNB} r_{cc}^{i}\left(\frac{1}{2}(2^{min(MinBE+i,MaxBE)} - 1)t_{bk} + t_{cca}\right) \tag{3.12}$$

where $MinBE$ and $MaxBE$ are the initial and maximum values of the back-off exponent respectively, t_{bk} is the duration of a back-off period and t_{cca} is the time of one CCA operation. Note that $MinBE$, $MaxBE$, t_{bk} and t_{cca} are constant values specified by the MAC protocols. On this basis, the idle listening time $t_{il}(r_{cc})$ can be formulated based on $t_{cc}(r_{cc})$ and it affects the number of overhearing bits $k_{oh}(r_{cc})$.

In addition, depending on the MAC protocols, the radio is either started once and kept active until the end of successful transmission, or it is turned off and restarted again with each retransmission. In other words, the radio startup times are either one or the average transmission times as shown by:

$$N_{st} = \begin{cases} 1 \\ N(r_{cc}, r_{cl}) \end{cases} \tag{3.13}$$

3.3.3 Validation of the Proposed Communication Energy Cost Model

In order to verify the validity of the proposed communication energy cost model, this section analyzes the communication energy consumption for the typical physical motes and compare the results with the reported measurements in previous works [82, 83]. In the two tests, different hardware components, CC2530 and CC2430, and protocol modes (non-beacon and beacon modes) are used to assess the flexibility of the proposed model.

Example 1: CC2530 + IEEE 802.15.4 non-beacon-enabled mode

In non-beacon-enabled networks, the receiver typically turns on its radio continuously, while the sender wakes up based on the application requirement. In this case, the communication process is always initiated by the sender. When it wants to transmit to the receiver, it first transmits a request command using unslotted CSMA/CA. The receiver responses with an acknowledge (ACK) frame and indicates whether there is pending data for it. To model and calculate the energy consumption in this process, this section refers to a simple application in [82], with only one sender and one receiver both based on CC2530. The communication process and the state diagram between the sender and the receiver without pending data are depicted in Fig. 3.3.

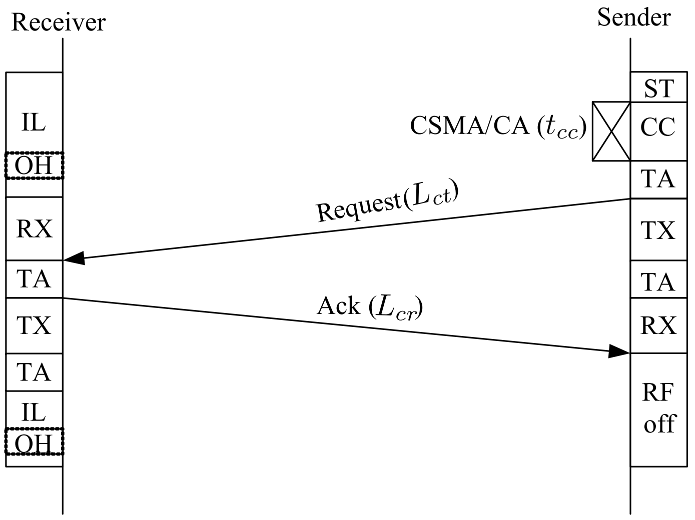


Figure 3.3: State diagram corresponding to one sender contacting one receiver without pending data in a non-beacon-enabled network.

According to [82, 84] and IEEE 802.15.4 non-beacon-enabled mode [85], the values of the parameters determined by the hardware and MAC protocols are listed in the uper portion of Table 3.3. The application related parameters are easily obtained (see lower part of Table 3.3). The sender accesses the channel successfully at the first time and transmits without collision, hence $N(r_{cc}, r_{cl})$ equals one. According to Eq. (3.12), $t_{cc}(r_{cc})$ is about 1.25 ms. From the communication process the idle listening time of the receiver $t_{il}(p_{cc}) \approx 3.7$ ms.

Using Eq. (3.8), it obtains that the average energy consumption of the sender is about 0.22 mJ, which is close to the measurement result of 0.25 mJ in [82]. The deviation is due to the slightly different transmitting and receiving power in our analysis and their experiments.

Table 3.3: Values of the related parameters in the proposed communication energy cost model when applying CC2530 with IEEE 802.15.4 non-beacon-enabled mode.

HW	e_{st}	50 μJ	Under different test conditions, the values vary. This is taken from the measurement
	P_{cc}	61.5 mW	Receive sensibility is -50 dBm
	$e_{tx}(d)$	0.344 μJ	Radio in TX mode, 1 dBm output power, CPU idle, 28.7 mA
	e_{rx}	0.246 μJ	Radio in RX mode, -50 dBm input power, CPU idle, 20.5 mA
	e_{ta}	8.64 μJ	Assuming 15 mA as the turnaround power
	P_{il}	61.5 mW	Radio in RX mode, -50 dBm input power, CPU idle
	e_{oh}	0.344 μJ	Radio in RX mode, -50 dBm input power, CPU idle
MAC	L_{cr}	18 bytes	The control packet transmitted by the sender
	L_{ct}	11 bytes	The control packet received by the sender
	N_{ta}	2-4	Depends on the scenario
	N_{st}	1	Only startup once
APP	L_d	0	Without data packet transmission
	T_{scd}	5 ms	Defined by the users
APP&MAC	$N(r_{cc}, r_{cl})$	1	Transmit only once
	r_{cc}	0	No channel access failure
	r_{cl}	0	No collision
	$t_{cc}(r_{cc})$	1.25 ms	The first access average time
	$t_{il}(r_{cc})$	3.7 ms	Idle listening time
	$L_{oh}(r_{cc})$	0	No overhearing

Example 2: CC2430 + IEEE 802.15.4 beacon-enabled mode

In beacon-enabled networks, the receiver will periodically wake up to broadcast a beacon that specifies the superframe structure and keep active during this superframe duration. When a sender wants to transmit data to the receiver in a beacon-enabled networks, it first listens for the network beacon. When the beacon is found, the sender transmits its data frame, using slotted CSMA/CA, to the receiver. The receiver can send an optional ACK to confirm the successful reception. To measure the energy consumption for both of the sender and receiver in a beacon-enabled IEEE 802.15.4 network, the experiment in [83] sets up a simple scenario with only one sender and one receiver both based on CC2430. Fig. 3.4 illustrates the communication procedure between the sender and receiver.

In this case, the duration of one superframe of the receiver is 15.36 ms; the beacon and the data frame are 26 bytes and 50 bytes, respectively. As in the last example, $N(p_{cc}, p_{cl}) = 1$ and $t_{cc}(p_7) \approx 1.25$ ms. During one superframe duration, the idle listening time of the receiver

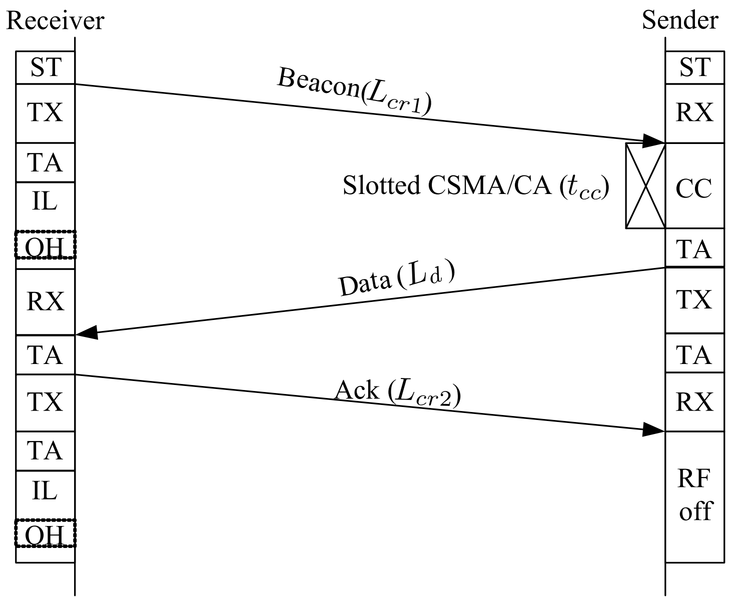


Figure 3.4: State diagram corresponding to one sender transmitting data to one receiver in a beacon-enabled network.

$t_{il}(p_{cc}) \approx 11.66\ ms$. Finally, the hardware and the MAC parameters are set according to [86, 87] and IEEE 802.15.4 beacon-enabled mode [85]; they are listed in Table 3.4.

The average energy consumption of the receiver calculated by Eq. (3.8) is about 1.284 mJ, while the measurement result in [83] is 1.368 mJ; the estimated and the measured cost for the sender are 0.53 mJ and 0.91 mJ respectively. The deviation is due to the uncertainty of the synchronization process that causes 0.32 mJ additional energy cost in the sender node and a small additional processing energy that has not considered in the model.

According to the comparison results of the above two examples, it indicates that the communication energy cost of WSN nodes by using the proposed model is very close to the realistic measurements.

3.4 Evaluation Metrics for Task Allocation Algorithms

This section introduces the metrics that are used to evaluate the performance of the task allocation approaches. The metrics mainly include: network lifetime, network energy consumption and residual energy distribution, and time requirement, in which the metric of network lifetime is chosen in this work.

Table 3.4: Values of the related parameters in the proposed communication energy cost model when applying CC2430 with IEEE 802.15.4 non-beacon-enabled mode.

HW	e_{st}	e_{st_r} =95 μJ e_{st_s} =185 μJ	Under different test conditions, the values vary. These are taken from the measurement
	P_{cc}	80.1 mW	Receive sensibility is -50 dBm
	$e_{tx}(d)$	0.323 μJ	Radio in TX mode, 0 dBm output power, low CPU activity, 26.9 mA
	e_{rx}	0.320 μJ	Radio in RX mode, -50 dBm input power, low CPU activity, 26.7 mA
	e_{ta}	10.368 μJ	Assuming 18 mA as the turnaround power [87]
	P_{il}	80.1 mW	Radio in RX mode, -50 dBm input power, low CPU activity
	e_{oh}	0.320 μJ	Radio in RX mode, -50 dBm input power, low CPU activity
MAC	L_{cr}	37 bytes	The control packet transmitted by the sender
	L_{ct}	0 bytes	The control packet received by the sender
	N_{ta}	2 or 3	The number of turnaround times is 2 and 3 for the sender and receiver respectively
	N_{st}	1	Only startup once
APP	L_d	50 bytes	Data packet length
	T_{scd}	15.36 ms	Defined by the users
APP&MAC	$N(r_{cc}, r_{cl})$	1	Transaction only happens once
	r_{cc}	0	No channel access failure
	r_{cl}	0	No collision
	$t_{cc}(r_{cc})$	1.25 ms	The average channel access duration
	$t_{il}(r_{cc})$	11.66 ms	Idle listening time
	$L_{oh}(r_{cc})$	0	No overhearing

Network Lifetime Definition

Due to the limited energy sources of WSN nodes, extending the network lifetime is one of the most important concerns for WSN applications. There are many parameters influencing network lifetime, e.g., the number of the alive sensor nodes in the network, the coverage and connectivity of the network, etc. [88]. Therefore, the precise definitions of the network lifetime (NL) vary in relevant literatures. The common definitions are presented as follows:

a) Defining NL according to the number of died sensor nodes: NL is defined as the time when k ($1 \leq k \leq n$) out the total n sensor nodes in the network die.

b) Defining NL according to the network coverage: When the region of interest is covered by less than k sensor nodes, the network ends.

c) Defining NL according to the network connectivity: The network is considered to die when there are less than k sensor nodes which have a path to the sink node.

d) Defining NL according to the quality of service: NL is defined as the time until the network cannot guarantee the application requirements.

Although there are so many different definitions of the network lifetime, the most frequently used definition in existing literatures is k out n sensor nodes die. Especially, when $k = 1$, which means the network lifetime is defined as the time when the first sensor node runs out of energy, has been widely used such as in [21, 29, 56, 31, 32, 42, 88, 89].

Network Energy Consumption and Residual Energy Distribution

In addition to the metric of network lifetime, the overall network energy consumption and residual energy distribution are also preferred by many researches to estimate the performances of the task allocation approaches. Several studies assume that a certain amount of energy can be used for the whole network. The network is out of functionality when this energy is completely depleted [73, 74]. Therefore, minimizing the summation of all sensor nodes' energy consumption (termed as network energy consumption) is the main objective of these studies. Besides, the standard deviation of the residual energy of each sensor node is used by a number of publications to evaluate the task allocation methods [59, 61, 90, 66]. A smaller value of the residual energy distribution represents more balanced lifetime of each sensor node, which indicates a better performance of the task allocation algorithms.

Time Requirement

In WSNs, the time requirement is another important metric that needs to be considered when estimating the task allocation algorithms [58, 31, 90, 66, 69, 73]. In many WSN applications, it is mandatory to quickly know the presence of some events to make a quick response. The time requirements vary for different applications. Typically, the makespan from the first task being executed to the last one being completed should be no longer than a prescribed time. Moreover, the time requirement should also be satisfied for specific sensor network communication protocols.

This work mainly uses the metric of network lifetime to estimate the performance of task allocation algorithms. It considers that each sensor node in the network is equally critical to the network and the energy exhaustion of any node will lead to the failure of the whole network. In other words, it uses the 1 out of *n* network lifetime definition when designing the task allocation algorithms.

3.5 Summary

This chapter has presented the system models used for the task allocation problem including the application models, the energy cost functions of WSN nodes and the network lifetime definition.

Typically, a specific WSN application is made up of a set of processing tasks and a paired set of communication tasks. According to this characteristic, directed acyclic graphs (DAGs) are used to model the applications. The static and dynamic applications are modeled by the standard DAG graph and DAG graph with conditional branches, respectively.

Next, this chapter formulates the energy cost functions of WSN nodes when they are executing the processing, communication and sleeping tasks. In order to accurately formulate the communication cost function, this work proposes a detailed communication cost model. It considers both the data packet transmission and the communication overhead activities including radio startup, channel access, etc. The proposed model is validated by comparing with the reported measurements in previous works [82, 83] based on CC2430 and CC2530 hardware platforms. The comparison results show that the communication cost by using the proposed model is very close to the realistic measurements.

Then, estimation metrics for task allocation algorithms are presented, which includes network lifetime, network energy consumption and residual energy distribution, and time requirement. Among the different metrics, the network lifetime is chosen by this work. The network lifetime is defined as the time until the first node runs out of energy, since each sensor node is considered to be equally critical for the network.

Based on the presented system models, the next chapters, Chapters 4 and 5, will describe the proposed task allocation algorithms.

4 Task Allocation Algorithms for Cluster-Based WSNs

4.1 Introduction **38**

4.2 System Model for CSTA, CDTA and DOOTA **38**

4.3 CSTA: Centralized Static Task Allocation Algorithm for Cluster-Based WSNs **39**

4.3.1 Modeling the Static Task Allocation Problem 39

4.3.2 CSTA Algorithm 40

4.4 CDTA: Centralized Dynamic Task Allocation Algorithm for Cluster-Based WSNs **44**

4.4.1 Modeling the Dynamic Task Allocation Problem 45

4.4.2 CDTA Algorithm 45

4.5 DOOTA: Distributed Optimal On-line Task Allocation Algorithm for Cluster-Based WSNs **48**

4.5.1 Analysis of the Optimal Task Allocation Solution 48

4.5.2 DOOTA Algorithm 50

4.6 Simulation Results **56**

4.6.1 Simulation Setup 56

4.6.2 Evaluations of CSTA, CDTA and DOOTA for Symmetric WSNs . . . 57

4.6.3 Evaluations of CSTA, CDTA and DOOTA for Asymmetric WSNs . . . 63

4.7 Summary **67**

4.1 Introduction

Achieving the fair balanced energy consumption among the sensor nodes to maximize the overall network lifetime is a primary concern in WSNs. To reach this objective, energy aware task allocation algorithms are proposed in this chapter.

Due to the simple routing characteristics and protocol demands, this chapter considers the cluster based hierarchical WSNs. The WSN applications are modeled by the Directed Acyclic Graphs (DAGs). Based on these models, a centralized static task allocation (CSTA) algorithm is firstly proposed in Section 4.3. It models the task allocation problem as partitioning the DAG graph into 2 parts and distributing to the slave and master nodes. By using a binary vector variable to represent the partition cut, CSTA algorithm for maximizing the network lifetime is formulated as a binary integer linear programming (BILP) problem. CSTA provides the static partition solutions which enables each slave node apply one fixed time invariant partition cut to balance the workload distribution of the tasks. Later on, Section 4.4 proposes a centralized dynamic task allocation (CDTA) algorithm based on linear programming (LP). It is executed in the gateway and achieves a more balanced workload distribution between the slave and master nodes by using multiple partition cuts with different weights. Moreover, Section 4.5 proves that the optimal task allocation solution for each slave and master nodes is made up of at most two partition cuts. According to this characteristic, a lightweight distributed optimal on-line task allocation (DOOTA) algorithm is further proposed. Afterwards, the proposed task allocation algorithms are evaluated by extensive simulation results in Section 4.6. At last, Section 4.7 summarizes this chapter.

4.2 System Model for CSTA, CDTA and DOOTA

The proposed task allocation algorithms in this chapter target at the cluster based hierarchical WSNs as shown in Section 2.2.1. Due to the simple routing characteristics and protocol demands, such hierarchical structure has been widely used in many industrial applications, e.g., tire pressure monitoring [91], industry carbon monoxide detection [92], smart home [12] and health care [93], etc. Moreover, this kind of structure efficiently increases the network scalability and lifetime for common applications [16, 41]. Given a WSN, it can be firstly grouped into numbers of clusters by clustering approaches such as LEACH [41]. The proposed task allocation algorithms are carried out within the cluster, since each cluster can work independently [42]. In addition, this work assumes a perfect time synchronization and a negligible inter-cluster interference in the network, which can be achieved by the time division multiple access (TDMA) based protocols as performed in [42, 69].

In cluster based WSNs, the slave nodes have to collaborate with the master node to complete the local static or dynamic applications individually. Typically, these applications may be either the same or varied. In this chapter, the static application is considered and each slave node executes the same application.

4.3 CSTA: Centralized Static Task Allocation Algorithm for Cluster-Based WSNs

This section firstly models the task allocation problem for the cluster based WSNs, and then proposes the centralized static task allocation (CSTA) algorithm based on the binary integer linear programming (BILP). Each cluster in the network applies the CSTA algorithm parallelly to balance the energy consumption of the slave and master nodes.

4.3.1 Modeling the Static Task Allocation Problem

In the cluster, each slave node completes its own application tasks with the collaboration of the master node individually. The static task allocation problem is equivalent to divide the modeled DAG graph into two subgraphs, $\boldsymbol{G}_s = (\boldsymbol{V}_s, \boldsymbol{E}_s)$ and $\boldsymbol{G}_m = (\boldsymbol{V}_m, \boldsymbol{E}_m)$, with a time invariant partition cut $\boldsymbol{X}$ as illustrated in Fig. 4.1. The amount of the transmitted data from the slave node to its master node is the summation of the weights of the edges which cross the cut $\boldsymbol{X}$. Taking Fig. 4.1 for example, the amount of the transmitted data equals $l(e_5) + l(e_6)$. As the workload is completed by the cooperation of the slave node and its master node, $\boldsymbol{G}_s$ and $\boldsymbol{G}_m$ are restricted to:

$$\begin{aligned} \boldsymbol{G}_s \cup \boldsymbol{G}_m &= \boldsymbol{G} \\ \boldsymbol{G}_s \cap \boldsymbol{G}_m &= \varnothing \\ \boldsymbol{G}_s &\neq \varnothing \\ \boldsymbol{G}_m &\neq \varnothing \end{aligned} \tag{4.1}$$

To guarantee that the data is transmitted from the slave to the master, each edge that crosses the partition cut has to satisfy:

$$\begin{aligned} src(e) &\in \boldsymbol{G}_s \\ snk(e) &\in \boldsymbol{G}_m \end{aligned} \tag{4.2}$$

where $src(e)$ and $snk(e)$ are the source and sink vertexes of edge e, respectively.

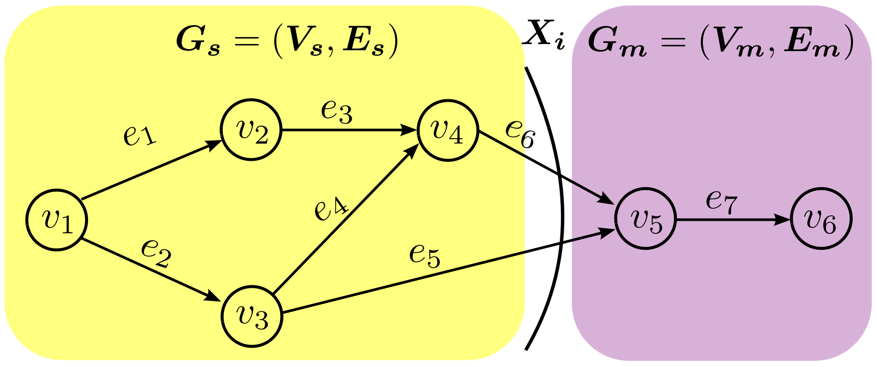


Figure 4.1: Schematic diagram of task allocation for local application.

4.3.2 CSTA Algorithm

The goal of CSTA algorithm is to extend the network lifetime by balancing the energy cost of the slave and the master nodes based on the BILP for both *asymmetric* and *symmetric* networks. It runs on the gateway which is typically assumed to be very powerful and without the energy limits.

Firstly, to formulate the task allocation problem as a BILP problem, an $K \times 1$ binary vector variable is used to represent the partition cut $\boldsymbol{X}$ of the application, i.e., $\boldsymbol{X} = [x(v_1), \cdots, x(v_k), \cdots, x(v_K)]^T$. K is the number of vertexes of the local DAG graph, and $x(v_k)$ is a boolean parameter which indicates whether the vertex v_k belongs to the slave node or the master node as the following definition:

$$x(v_k) = \begin{cases} 1, & \text{if } v_k \text{ belongs to the slave node} \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

Based on this vector variable, the constraints for guaranteeing the collaboration of the slave and master nodes, Eq. (4.1), and the direction of the transmitted data, Eq. (4.2), can be represented as the following matrix expressions:

$$1 \leq \mathbf{1}_{1\times K}\boldsymbol{X} \leq K - 1 \tag{4.4}$$

$$\boldsymbol{B}^T\boldsymbol{X} \geqq \mathbf{0} \tag{4.5}$$

where $\mathbf{1}_{1\times K}$ is an $1 \times K$ all one vector; $\mathbf{0}$ is an all zero vector; $\boldsymbol{B}^T$ is the transpose of the incidence matrix of the DAG graph, $\boldsymbol{B}$, which has the row for each vertex and column for each edge: $\boldsymbol{B}(v, e)$ equals 1 if edge e leaves vertex v, -1 if edge e enters v and 0 otherwise. Let $l_n(v_k)$ denote the net generated data of task v_k in the DAG, it is formed as:

$$l_n(v_k) = \sum_{e \in out(v_k)} l(e) - \sum_{e \in in(v_k)} l(e) \tag{4.6}$$

where $in(v_k)$ and $out(v_k)$ are the input and output edges of vertex v_k. Let $\boldsymbol{L}_n = [l_n(v_1), \cdots, l_n(v_K)]$ denote the net generated data of all the vertexes in the DAG, the amount of transmitted data from the slave node to the master node, L, can be expressed as:

$$L = \boldsymbol{L}_n \boldsymbol{X} \tag{4.7}$$

Based on the above vector parameters, the energy consumption of the slave node and the master node are then formulated as the linear functions of the partition cut $\boldsymbol{X}$. For a given partition cut $\boldsymbol{X}_i$: the energy cost of the slave node i, $E_i(\boldsymbol{X}_i)$, consists of cost of executing the assigned tasks and transmitting the data to the master node; the energy cost of the master node, $E_{mi}(\boldsymbol{X}_i)$ is made up of receiving the data from slave node i and processing the received data. Combining Eqs. (3.2) to (3.4) and (4.7), $E_i(\boldsymbol{X}_i)$ and $E_{mi}(\boldsymbol{X}_i)$ can be formulated as:

$$E_i(\boldsymbol{X}_i) = \boldsymbol{E}_{pi}\boldsymbol{X}_i + e_o + e_{tx}(d_i)\boldsymbol{L}_n\boldsymbol{X}_i \tag{4.8}$$

$$E_{mi}(\boldsymbol{X}_i) = \boldsymbol{E}_{pm}(\mathbf{1}_{K\times 1} - \boldsymbol{X}_i) + e_o + e_{rx}\boldsymbol{L}_n\boldsymbol{X}_i \tag{4.9}$$

where $\boldsymbol{E}_{pi} = [E_{pi}(v_1), \cdots, E_{pi}(v_k), \cdots, E_{pi}(v_K)]$ and $\boldsymbol{E}_{pm} = [E_{pm}(v_1), \cdots, E_{pm}(v_k), \cdots, E_{pm}(v_K)]$ represent the processing energy cost of each vertex when they are executed by the slave node i and the master node respectively. The corresponding execution times of slave node i and the master node, $T_i(\boldsymbol{X}_i)$ and $T_m(\boldsymbol{X}_i)$, are:

$$T_i(\boldsymbol{X}_i) = \boldsymbol{T}_{pi}\boldsymbol{X}_i + t_o + t_{tx}\boldsymbol{L}_n\boldsymbol{X}_i \tag{4.10}$$

$$T_m(\boldsymbol{X}_i) = \boldsymbol{T}_{pm}(\mathbf{1}_{K\times 1} - \boldsymbol{X}_i) + t_o + t_{rx}\boldsymbol{L}_n\boldsymbol{X}_i \tag{4.11}$$

where $\boldsymbol{T}_{pi} = [t_i(v_1), \cdots, t_i(v_k), \cdots, t_i(v_K)]$ and $\boldsymbol{T}_{pm} = [t_m(v_1), \cdots, t_m(v_k), \cdots, t_m(v_K)]$ stand for the time duration of each vertex when it is executed by the slave node i and the master node, respectively; and t_o is time consumption of the overhead activities in communications.

As the master node is in charge of its n slave nodes, it has to iterate n times to complete the applications for each of the slave nodes in one scheduling round. Thus, the master's energy cost and time consumption in one round, $E_m(\boldsymbol{X}_{1,\cdots,n})$ and $T_m(\boldsymbol{X}_{1,\cdots,n})$, are:

$$E_m(\boldsymbol{X}_{1,\cdots,n}) = \sum_{i=1}^{n} \big(\boldsymbol{E}_{pm}(\mathbf{1}_{K\times 1} - \boldsymbol{X}_i) + e_o + e_{rx}\boldsymbol{L}_n\boldsymbol{X}_i\big) \tag{4.12}$$

$$T_m(\boldsymbol{X}_{1,\cdots,n}) = \sum_{i=1}^{n} \big(\boldsymbol{T}_{pm}(\mathbf{1}_{K\times 1} - \boldsymbol{X}_i) + t_o + t_{rx}\boldsymbol{L}_n\boldsymbol{X}_i\big) \tag{4.13}$$

where $\mathbf{1}_{K\times 1}$ is an $K \times 1$ all one vector.

Let T_{scd} denote the time duration of one scheduling round, which is set up by the users according to different WSN applications. According to Eqs. (4.10) and (4.13), the sleeping time of slave node i, $t_{slp\_i}(\boldsymbol{X}_i)$ and $t_{slp\_m}(\boldsymbol{X}_{1,\cdots,n})$, can be formulated as:

$$t_{slp\_i}(\boldsymbol{X}_i) = T_{scd} - \big(\boldsymbol{T}_{pi}\boldsymbol{X}_i + t_o + t_{tx}\boldsymbol{L}_n\boldsymbol{X}_i\big) \tag{4.14}$$

$$t_{slp\_m}(\boldsymbol{X}_{1,\cdots,n}) = T_{scd} - \sum_{i=1}^{n} \big(\boldsymbol{T}_{pm}(\mathbf{1}_{K\times 1} - \boldsymbol{X}_i) + t_o + t_{rx}\boldsymbol{L}_n\boldsymbol{X}_i\big) \tag{4.15}$$

Correspondingly, the slave energy cost of slave node i and the master node, $E_{slp\_i}(\boldsymbol{X}_i)$ and $E_m(\boldsymbol{X}_{1,\cdots,n})$, are:

$$\begin{aligned} E_{slp\_i}(\boldsymbol{X}_i) &= P_{slp\_i} t_{slp\_i}(\boldsymbol{X}_i) \\ &= P_{slp\_i}\Big(T_{scd} - \big(\boldsymbol{T}_{pi}\boldsymbol{X}_i + t_o + t_{tx}\boldsymbol{L}_n\boldsymbol{X}_i\big)\Big) \end{aligned} \tag{4.16}$$

$$\begin{aligned} E_{slp\_m}(\boldsymbol{X}_{1,\cdots,n}) &= P_{slp\_m} t_{slp\_m}(\boldsymbol{X}_{1,\cdots,n}) \\ &= P_{slp\_m}\Big(T_{scd} - \sum_{i=1}^{n} \big(\boldsymbol{T}_{pm}(\mathbf{1}_{K\times 1} - \boldsymbol{X}_i) + t_o + t_{rx}\boldsymbol{L}_n\boldsymbol{X}_i\big)\Big) \end{aligned} \tag{4.17}$$

After formating the energy cost of the slave and master nodes as the linear functions of the partition cut $\boldsymbol{X}_i$, the problem of maximizing the network lifetime using CSTA is formulated as a BILP problem for both the *asymmetric* and *symmetric* WSNs.

For the *asymmetric* WSNs, the battery energy and the transmitting power of the slave nodes are varied. Correspondingly, they may use different partition cuts. Since the network lifetime, NL, is defined as the time until the first node dies, given an asymmetric network with 1 master node and n slave nodes, NL can be expressed as:

$$NL = min\Big\{\frac{Bat_m}{E_m(\boldsymbol{X}_{1,\cdots,n}) + E_{slp\_m}(\boldsymbol{X}_{1,\cdots,n})}, \frac{Bat_1}{E_i(\boldsymbol{X}_i) + E_{slp\_i}(\boldsymbol{X}_i)}\Big\}, i = 1, \cdots, n. \tag{4.18}$$

where Bat_m and Bat_i are the battery energy of the master node and the slave nodes, respectively.

Consequently, maximizing NL is equivalent to minimize $\frac{1}{NL}$, which can be formulated as a BILP problem as follows:

$$\arg\min_{\boldsymbol{X}_i} \; max\left\{\frac{E_m(\boldsymbol{X}_{1,\cdots,n}) + E_{slp\_m}(\boldsymbol{X}_{1,\cdots,n})}{Bat_m},\; \frac{E_i(\boldsymbol{X}_i) + E_{slp\_i}(\boldsymbol{X}_i)}{Bat_i}\right\}, i = 1, \cdots, n \tag{4.19}$$

subject to:

$$1 \leq \mathbf{1}_{1\times K}\boldsymbol{X}_i \leq K - 1$$
$$\boldsymbol{B}^T\boldsymbol{X}_i \geqq \mathbf{0}$$

In contrast to the *asymmetric* WSNs, the slave nodes use the same partition cut in the *symmetric* networks, since they are typically assumed to have the same battery energy, transmitting and processing power. According to Eqs. (4.8) and (4.12)The energy cost of the slave and master nodes for executing the applications, $E_s(\boldsymbol{X})$ and $E_m(\boldsymbol{X})$, can be reformed as:

$$E_s(\boldsymbol{X}) = \boldsymbol{E}_{ps}\boldsymbol{X} + e_o + e_{tx}\boldsymbol{L}_n\boldsymbol{X} \tag{4.20}$$

$$E_m(\boldsymbol{X}) = n\big(\boldsymbol{E}_{pm}(\mathbf{1}_{K\times 1} - \boldsymbol{X}) + e_o + e_{rx}\boldsymbol{L}_n\boldsymbol{X}\big) \tag{4.21}$$

Based on Eqs. (4.16) and (4.17), the sleeping energy cost of the slave and master nodes in one scheduling period are:

$$\begin{aligned} E_{slp\_s}(\boldsymbol{X}) &= P_{slp\_s}t_{slp\_s}(\boldsymbol{X}) \\ &= P_{slp\_s}\Big(T_{scd} - \big(\boldsymbol{T}_{ps}\boldsymbol{X} + t_o + t_{tx}\boldsymbol{L}_n\boldsymbol{X}\big)\Big) \end{aligned} \tag{4.22}$$

$$\begin{aligned} E_{slp\_m}(\boldsymbol{X}) &= P_{slp\_m}t_{slp\_m}(\boldsymbol{X}) \\ &= P_{slp\_m}\Big(T_{scd} - n\big(\boldsymbol{T}_{pm}(\mathbf{1}_{K\times 1} - \boldsymbol{X}) + t_o + t_{rx}\boldsymbol{L}_n\boldsymbol{X}\big)\Big) \end{aligned} \tag{4.23}$$

Correspondingly, maximizing NL is equivalent to minimize the energy cost of the slave and master nodes, which can be formulated as the following BILP format:

$$\arg\min_{\boldsymbol{X}} \; max\{E_m(\boldsymbol{X}) + E_{slp\_m}(\boldsymbol{X}),\; E_s(\boldsymbol{X}) + E_{slp\_s}(\boldsymbol{X})\} \tag{4.24}$$

subject to:

$$1 \leq \mathbf{1}_{1\times K}\boldsymbol{X} \leq K - 1,$$
$$\boldsymbol{B}^T\boldsymbol{X} \geqq \mathbf{0}$$

Note that, the partition cut $\boldsymbol{X}_i$ for the asymmetric networks and the partition cut $\boldsymbol{X}$ for the symmetric networks are time invariant binary vectors. They are used to distribute the tasks statically for the slave and master nodes. Under this static strategy, the optimal static partition solutions can be obtained by solving the above Eqs. (4.19) and (4.24). By applying the obtained partition cuts, the network lifetime can be efficiently extended. Moreover, the complexity of CSTA for *asymmetric* networks is related to the number of slave nodes, since the slave nodes are considered to apply different partition cuts. For *symmetric* networks, the complexity of CSTA does not change due to the fact that the slave nodes use the same partition cut. This characteristic will be validated by the simulation results inSection 4.6.2.

Instead of using the static partition solutions, the dynamic task allocation by using multiple partition cuts can achieve a more balanced workload distribution between the slave and master nodes, thereby extending the network lifetime longer. To this end, the next section illustrates a dynamic task allocation algorithm by using multiple partition cuts with different weights.

4.4 CDTA: Centralized Dynamic Task Allocation Algorithm for Cluster-Based WSNs

Dynamic task allocation can achieve more balanced energy consumption than using the static strategy. Let us firstly consider a simple WSN scenario: only one slave node and one master node collaborate to complete one application in the network. They have the same battery energy and the application DAG graph is illustrated in Fig. 4.2. In the DAG, executing vertex v_2 consumes much more energy than v_1 and v_3 for both the slave and master nodes. Obviously, applying either partition cut 1 or partition cut 2 will lead the master node or the slave node overloaded and die soon, which makes the network work in very short time. While alternatively applying these two partition cuts for the slave and master nodes to share the heavy workload of vertex v_2 can achieve more balanced workload distribution, thereby increasing the network lifetime.

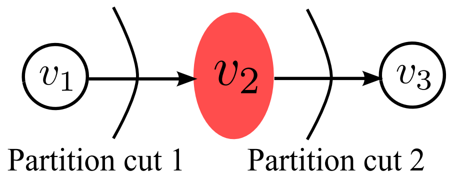


Figure 4.2: Schematic diagram of the dynamic task allocation using multiple partition cuts.

This section extends the CSTA algorithm to a centralized dynamic task allocation (CDTA) algorithm by using multiple partition cuts with the corresponding weights. It firstly models the dynamic task allocation problem, and then presents the CDTA algorithm for both asymmetric and symmetric networks.

4.4.1 Modeling the Dynamic Task Allocation Problem

Motivated by the above example, it is considered in this section that the slave node i applies different partition cut in each DAG execution round. Since the sleeping cost is typically very small that can be neglected, this section mainly focuses on the energy cost by executing the applications. Let $\boldsymbol{X}_i^j$ denote the partition cut that slave node i exploits in the j-th scheduling round. Assuming the network dies after T rounds, the problem of maximizing the network is then to find $\boldsymbol{X}_i^j$, $i = 1, \cdots, n$ and $j = 1, \cdots, T$, to maximize T, which can be formulated as:

$$\arg\max_{\boldsymbol{X}_i^j} \quad T \tag{4.25}$$

subject to:

$$\sum_{j=1}^{T} E_i(\boldsymbol{X}_i^j) \leq Bat_i, \; i = 1, \cdots, n; j = 1, \cdots, T$$

$$\sum_{j=1}^{T} E_m(\boldsymbol{X}_1^j, \cdots, \boldsymbol{X}_n^j) \leq Bat_m$$

$$1 \leq \mathbf{1}_{1\times K}\boldsymbol{X}_i \leq K - 1$$

$$\boldsymbol{B}^T\boldsymbol{X}_i \geqq \mathbf{0}$$

Typically, the network lasts hundreds of thousands DAG execution rounds, which means T is a very huge number. It is very complicated for the dynamic task allocation to calculate the nT variables. Even for the symmetric networks in which all the slave nodes use the same partition solution, there are still T variables.

To make this complex problem feasible, the CDTA is proposed in the next section by integrating the T binary vector variables as a probability vector variable for each slave node.

4.4.2 CDTA Algorithm

The proposed CDTA algorithm reduces the complexity and formulates the dynamic task allocation problem as a LP problem for the asymmetric and symmetric networks, respectively.

According to Eqs. (4.8) and (4.12), the energy cost of slave node i and the master node in the j-th DAG execution round, $E_i(\boldsymbol{X}_i^j)$, $E_m(\boldsymbol{X}_1^j, \cdots, \boldsymbol{X}_n^j)$, can be reformed as:

$$E_i(\boldsymbol{X}_i^j) = \boldsymbol{E}_{pi}\boldsymbol{X}_i^j + e_o + e_{tx}(d_i)\boldsymbol{L}_n\boldsymbol{X}_i^j \tag{4.26}$$

$$E_m(\boldsymbol{X}_1^j, \cdots, \boldsymbol{X}_n^j) = \sum_{i=1}^{n} \left(\boldsymbol{E}_{pm}(\mathbf{1}_{K\times 1} - \boldsymbol{X}_i^j) + e_o + e_{rx}\boldsymbol{L}_n\boldsymbol{X}_i^j\right) \tag{4.27}$$

Assuming the network dies after T rounds, the average energy cost of slave node i and the master node, $\overline{E}_i$ and $\overline{E}_m$, are:

$$\begin{aligned}\overline{E}_i &= \frac{1}{T}\sum_{j=1}^{T} E_i(\boldsymbol{X}_i^j) \\ &= e_o + \big(\boldsymbol{E}_{pi} + e_{tx}(d_i)\boldsymbol{L}_n\big)\frac{1}{T}\sum_{j=1}^{T}\boldsymbol{X}_i^j \end{aligned} \tag{4.28}$$

$$\begin{aligned}\overline{E}_m &= \frac{1}{T}\sum_{j=1}^{T} E_m(\boldsymbol{X}_1^j, \cdots, \boldsymbol{X}_n^j) \\ &= \sum_{i=1}^{n}\Big(\boldsymbol{E}_{pm}\mathbf{1}_{K\times 1} + e_o + (e_{rx}\boldsymbol{L}_n - \boldsymbol{E}_{pm})\frac{1}{T}\sum_{j=1}^{T}\boldsymbol{X}_i^j\Big) \end{aligned} \tag{4.29}$$

Let $\boldsymbol{\chi}_i = \frac{1}{T}\sum_{j=1}^{T}\boldsymbol{X}_i^j = [\chi_i(v_1), \cdots \chi_i(v_k), \cdots \chi_i(v_K)]^T$, in which each element $\chi_i(v_k)$ satisfies:

$$\chi_i(v_k) = \frac{1}{T}\sum_{j=1}^{T} x_i^j(v_k)$$

Due to the fact that $x_i^j(v_k)$ only could be either 0 or 1, $\chi_i(v_k)$ can be treated as a probability number between 0 and 1. It indicates how often the vertex v_k is executed by slave node i. By using $\boldsymbol{\chi}_i$, Eqs. (4.28) and (4.29) can be expressed as:

$$\overline{E}_i(\boldsymbol{\chi}_i) = e_o + \big(\boldsymbol{E}_{pi} + e_{tx}(d_i)\boldsymbol{L}_n\big)\boldsymbol{\chi}_i \tag{4.30}$$

$$\overline{E}_m(\boldsymbol{\chi}_1, \cdots, \boldsymbol{\chi}_n) = \sum_{i=1}^{n}\big(\boldsymbol{E}_{pm}\mathbf{1}_{K\times 1} + e_o + (e_{rx}\boldsymbol{L}_n - \boldsymbol{E}_{pm})\boldsymbol{\chi}_i\big) \tag{4.31}$$

Note that the parameters in Eqs. (4.30) and (4.31) are constant values except $\boldsymbol{\chi}_i$. Correspondingly, the constraints for guaranteeing the collaboration of the slave and master nodes and the direction of the transmitted data, Eqs. (4.4) and (4.5), could be reformed as:

$$1 \leq \mathbf{1}_{1\times K}\,\boldsymbol{\chi}_i \leq K - 1 \tag{4.32}$$

$$\boldsymbol{B}^T\boldsymbol{\chi}_i \geqq \mathbf{0} \tag{4.33}$$

Given an asymmetric WSN which contains one master node and n slave nodes, based on

Eqs. (4.30) and (4.31), NL can be expressed as:

$$NL = min\Big\{\frac{Bat_m}{\overline{E}_m(\chi_1,\cdots,\chi_n)}, \frac{Bat_1}{\overline{E}_1(\chi_1)}, \cdots, \frac{Bat_i}{\overline{E}_i(\chi_i)}, \cdots, \frac{Bat_n}{\overline{E}_n(\chi_n)}\Big\} \tag{4.34}$$

Consequently, the problem of maximizing NL is formulated as the following LP format:

$$\begin{aligned} &\arg\min_{\chi_i}\ max\Big\{\frac{\overline{E}_m(\chi_1,\cdots,\chi_n)}{Bat_m}, \frac{\overline{E}_1(\chi_1)}{Bat_1}, \cdots, \frac{\overline{E}_i(\chi_i)}{Bat_i}, \cdots, \frac{\overline{E}_n(\chi_n)}{Bat_n}\Big\} \\ &\text{subject to:} \\ &\qquad 1 \le \mathbf{1}_{1\times K}\chi_i \le K-1, \quad i = 1,\cdots,n \\ &\qquad \boldsymbol{B}^T\chi_i \geqq \mathbf{0} \end{aligned} \tag{4.35}$$

Similar to the CSTA algorithm in Section 4.3, the average energy cost of the slave and master nodes for symmetric networks, $\overline{E}_s(\chi)$ and $\overline{E}_m(\chi)$, can be reformed as:

$$\overline{E}_s(\chi) = e_o + \big(\boldsymbol{E}_{ps} + e_{tx}\boldsymbol{L}_n\big)\chi \tag{4.36}$$

$$\overline{E}_m(\chi) = n\big(\boldsymbol{E}_{pm}\mathbf{1}_{K\times 1} + e_o + (e_{rx}\boldsymbol{L}_n - \boldsymbol{E}_{pm})\chi\big) \tag{4.37}$$

Maximizing NL is equivalent to minimize the average energy cost of the slave and master nodes, which is formulated as the following LP format:

$$\begin{aligned} &\arg\min_{\chi}\ max\{\overline{E}_m(\chi),\ \overline{E}_s(\chi)\} \\ &\text{subject to:} \\ &\qquad 1 \le \mathbf{1}_{1\times K}\chi \le K-1, \\ &\qquad \boldsymbol{B}^T\chi \geqq \mathbf{0} \end{aligned} \tag{4.38}$$

By solving the above LP problem, Eqs. (4.35) and (4.38), the partition solutions for the asymmetric and symmetric networks can be obtained. Note that, the obtained partition solutions, χ_i and χ, are made up of multiple partition cuts with the corresponding weights. Since the elements in each partition cut are either 1 or 0, the number of unique nonzero elements in χ determines the number of the partition cuts. E.g., if χ is a binary vector which has only one unique element 1, there is only one partition cut with the weight of 100%. According to this characteristic, Given a specific χ, it can be calculated by Algorithm 1. Taking the DAG graph in Fig. 4.2 for example, if LP returns $\chi_i = [1, 0.4, 0]^T = 0.4[1, 1, 0]^T + 0.6[1, 0, 0]^T$, it indicates that partition cut 2 and partition cut 1 are selected with the weights 40% and 60%, respectively.

Algorithm 1 Calculating the partition cuts and their weights

```
1:  Input: $\chi$
2:  Output: $X_1, \cdots ; p_1, \cdots$          % partition cuts and their weights
3:  $ID$ = find ($\chi \neq 0$);                 % find the nonzero elements in $\chi$
4:  $NZ$ = unique($\chi(ID)$);                   % find the elements which are unique
5:  $p_0 = 0$;
6:  for i=1:length($NZ$) do
7:      $X_i = \chi$;                            % partition cut
8:      $X_i(X_i \geq NZ(i)) = 1$;
9:      $X_i(X_i < NZ(i)) = 0$;
10:     $p_i = NZ(i) - p_0$;                     % the weights of partition cut $X_i$
11:     $p_0 = NZ(i)$;
12: end for
```

4.5 DOOTA: Distributed Optimal On-line Task Allocation Algorithm for Cluster-Based WSNs

The centralized algorithms typically suffer from computational complexity and also need to frequently collect the updates of the parameters from the sensor nodes to adapt to network changes, which are hard to achieve on-line and incur large delay. Due to the drawbacks of the centralized algorithms, this section proposes a distributed optimal on-line task allocation (DOOTA) algorithm. It firstly thoroughly analyzes the composition of the optimal task allocation solution. Based on the analysis, the very lightweight DOOTA algorithm is presented in details.

4.5.1 Analysis of the Optimal Task Allocation Solution

According to the task allocation solutions obtained by CDTA, a remarkable observation is found that the optimal task allocation solution always consists of two partition cuts. In order to provide the theoretical support, this section presents a in-depth analysis of the composition of the optimal task allocation solution, which also lays the foundation to the proposed DOOTA algorithm.

Given a specific application for the cluster based WSNs, it can be modeled as a DAG graph as introduced in Section 3.2. The number of the vertexes in the modeled DAG graph is limited. Since the partition cuts are made up of different combinations of the vertexes, the valid partition cuts which satisfy the collaboration and data direction constraints Eqs. (4.4) and (4.5) are countable. Each of the valid partition cuts maps to the corresponding energy cost of slave node i and its master node, i.e., E_i and E_{mi}. Considering a two dimension coordinate, on which the horizontal and vertical axises represent E_i and E_{mi}, respectively, the corresponding energy consumptions of all of the valid partition cuts and their combinations construct a convex set as

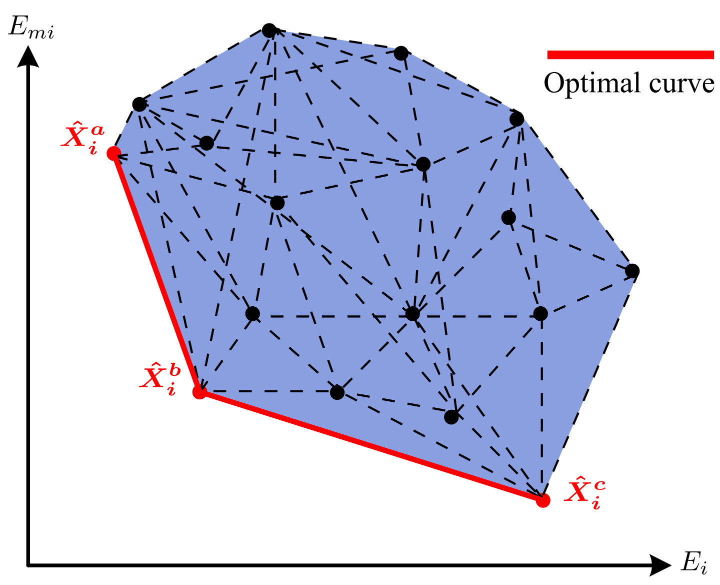


Figure 4.3: All combinations of the valid partition cuts and the corresponding energy consumption of slave node i and the master node.

shown in Fig. 4.3.

The smaller the energy consumption of the node, the longer it can survive. Thus, we target at the points that minimize the energy consumption of slave node i under the same energy consumption of the master node and vice versa. These points correspond to a subset of the boundary of the convex set, which is called *optimal curve* (see Fig. 4.3). It is actually a convex curve. Its start and end points, $\hat{X}_i^a$ and $\hat{X}_i^c$, stand for the minimum energy cost of the slave node i and the master, respectively. Given an arbitrary point in the convex set, i.e., one arbitrary possible partition solution, there exists at least one energy point that both the energy cost of slave node i and the master node are smaller than or equal to the given point on the *optimal curve*. In other words, there is always at least one partition solution on this *optimal curve* that makes the lifetimes of both the slave and master nodes longer than or equal to other partition solutions in the convex set. Therefore, the optimal solution is definitely on this *optimal curve*. Since the *optimal curve* is formed by the *important partition cuts* and the pair linear combinations of them, the optimal partition solution is made up of either one of the *important partition cuts* or two of them with the corresponding weights. Taking Fig. 4.3 for example, the optimal solution is either one of the *important partition cuts*, $\hat{\boldsymbol{X}}_i^a$, $\hat{\boldsymbol{X}}_i^b$ and $\hat{\boldsymbol{X}}_i^c$, or the linear combinations of $\hat{\boldsymbol{X}}_i^a$ and $\hat{\boldsymbol{X}}_i^b$ or $\hat{\boldsymbol{X}}_i^b$ and $\hat{\boldsymbol{X}}_i^c$.

Therefore, it only needs to find the *important partition cuts* and construct the *optimal curve* to obtain the optimal solution.

4.5.2 DOOTA Algorithm

The proposed DOOTA algorithm is actually an on-line negotiation among the slave and master nodes based on the *optimal curve* of each slave node. This section firstly illustrates how to calculate the *optimal curve* by using the binary decision diagram (BDD) theory in the off-line preparation stage, and then presents the detailed DOOTA algorithm.

Off-line Preparation

Typically, before deploying a network, the application is known in advance and can be easily modeled as a specific DAG graph. It is able to obtain the *important partition cuts* and formulate the *optimal curve* by off-line calculation. The off-line preparation consists of two steps: firstly calculating the valid partition cuts of the modeled DAG graph which satisfy the constraints Eqs. (4.4) and (4.5); and then obtaining the *important partition cuts* and formulating the *optimal curve*.

The complexity of calculating the valid partition cuts by enumeration for slave node i exponentially increases with the number of the vertexes in the DAG graph. For instance, given a DAG with L vertexes, the number of the possible partition cuts could be 2^L, which means the complexity of finding the valid partition cuts by enumeration is $O(2^L)$. By using a recursive boolean description of the problem and implementing the computations with BDDs, it is much more efficient to calculating the valid partition cuts. A BDD is a data structure which is used to represent a boolean function [94]; it can be also considered as a compressed representation of sets or relations. In this work, the partition cuts are made up of L boolean variables, since each vertex of the DAG graph belongs to either the slave or master. Thus, compressing the set of valid partition cuts can be achieved by using a BDD with L variables.

The key idea is to use the implicit partial order defined by the DAG graph. There may exist different valid paths from vertex v_l to the end vertex v_L in the modeled DAG graph. According to *constraint* Eqs. (4.4) and (4.5), vertex v_l cannot be assigned to the master if its destination vertexes belong to the slave node. Taking Fig. 4.1 as an example, there are two paths from v_3 to v_6: $v_3 \longrightarrow v_4 \longrightarrow v_5 \longrightarrow v_6$ and $v_3 \longrightarrow v_5 \longrightarrow v_6$. When v_3 is assigned to the master node, its destinations v_4 and v_5 must also belong to the master node. To formulate the BDD problem, v_l and $\bar{v}_l$ are used to represent that vertex v_l is distributed to the master and slave nodes, respectively. Let Ψ_l denote a boolean function of vertexes on the paths from v_l to v_L which returns true when v_l belongs to the master and *constraint* Eqs. (4.4) and (4.5) is satisfied. It can be expressed recursively as:

$$\Psi_l = v_l \bigwedge_{v_j \in dest(v_l)} \Psi_j$$

where $dest(v_l)$ is the set of the destination vertexes of v_l. Further on, when vertex v_l is assigned to the slave node i, its destination vertexes do not affect its assignment. Thus, boolean function, ψ_l, describing the paths satisfying *constraint* Eqs. (4.4) and (4.5) and starting at vertex v_l is:

$$\psi_l = \Psi_l \vee \left(\bar{v_l} \bigwedge_{v_j \in dest(v_l)} \psi_j \right)$$

According to collaboration constraint of the slave and master nodes, there exists $\psi_L = \Psi_L = v_L$ and $\Psi_1 = \emptyset$. Through the reversed iterative computations from ψ_L to ψ_1, the representation of the valid partition cuts ψ_1 can be obtained. However, the complexity may exponentially increase with L. Therefore, efficient ways to obtain the valid partition cuts are needed. This work uses the CUDD package [95], which can efficiently manipulate BDDs of boolean functions, to create the final BDD of ψ_1. When obtaining the BDD, the valid partition cuts are actually the paths from the top vertex to terminal 1. For instance, the BDD with the reversed order of the DAG graph in Fig. 4.1 is created as shown in Fig. 4.4. The dotted and solid edges represent that their source vertexes belong to the slave and master nodes, respectively. We can easily obtain that there are 6 paths from the vertex v_6 to terminal 1. Each path corresponds one valid partition cut that represents the distribution result of each vertex.

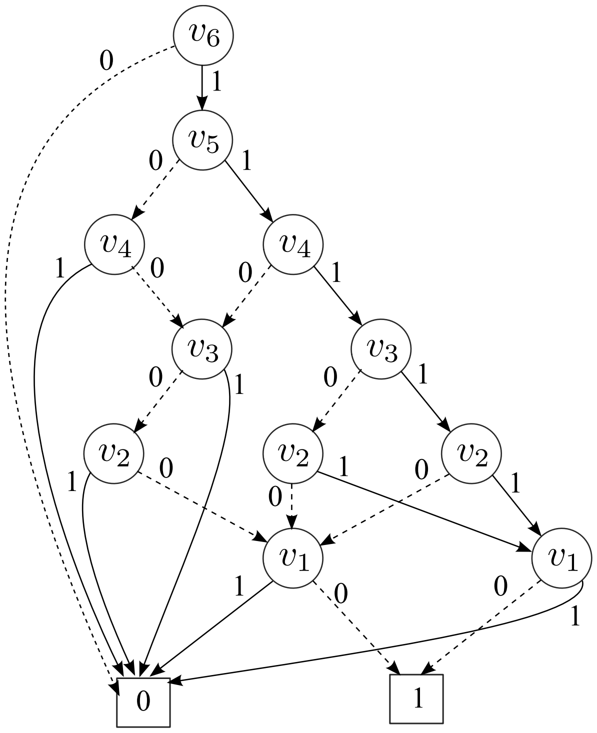


Figure 4.4: BDD graph of the DAG graph in Section 4.3.1.

After calculating the valid partition cuts, the off-line preparation then focuses on obtaining the *important partition cuts* and formulating the *optimal curve*. The energy consumptions of slave node i and the master node by using the valid partition cuts can be obtained according to Eqs. (4.8) and (4.9). The corresponding energy consumptions construct a convex set as Fig. 4.3 shows. The energy point that makes the slave node i consume the least energy is selected as the first important point (*important partition cut*). Afterwards, the slopes of the segments which start from the first important point to the others are calculated, in which the slope with the minimum value is selected as the second important point. Then, it starts from the current important point to find the next one until the point which corresponds to the least energy cost of the master node is found. The computation complexity depends on the specific energy points. In the worst case, it is $O(\frac{N_{vp}(N_{vp}-1)}{2})$, where N_{vp} is the number of the valid partition cuts.

Then, the *optimal curve* can be formulated based on the obtained *important partition cuts*. Taking Fig. 4.3 for example, the obtained *important partition cuts* are $\hat{\boldsymbol{X}}_i^a$, $\hat{\boldsymbol{X}}_i^b$ and $\hat{\boldsymbol{X}}_i^c$. The relation between E_{mi} and E_i on the *optimal curve* can be formulated as (4.39), which will be stored in slave node i before deploying it into the network.

$$E_{mi} = \begin{cases} A_0 E_i + B_0, & \text{if} E_i \in \left[E_i(\hat{\boldsymbol{X}}_i^a), E_i(\hat{\boldsymbol{X}}_i^b)\right] \\ A_1 E_i + B_1, & \text{if} E_i \in \left[E_i(\hat{\boldsymbol{X}}_i^b), E_i(\hat{\boldsymbol{X}}_i^c)\right] \end{cases} \tag{4.39}$$

where

$$A_0 = \frac{E_m(\hat{\boldsymbol{X}}_i^a) - E_m(\hat{\boldsymbol{X}}_i^b)}{E_i(\hat{\boldsymbol{X}}_i^a) - E_i(\hat{\boldsymbol{X}}_i^b)}, A_1 = \frac{E_m(\hat{\boldsymbol{X}}_i^b) - E_m(\hat{\boldsymbol{X}}_i^c)}{E_i(\hat{\boldsymbol{X}}_i^b) - E_i(\hat{\boldsymbol{X}}_i^c)},$$

$$B_0 = \frac{E_i(\hat{\boldsymbol{X}}_i^a)E_m(\hat{\boldsymbol{X}}_i^b) - E_m(\hat{\boldsymbol{X}}_i^a)E_i(\hat{\boldsymbol{X}}_i^b)}{E_i(\hat{\boldsymbol{X}}_i^a) - E_i(\hat{\boldsymbol{X}}_i^b)} \quad \text{and}$$

$$B_1 = \frac{E_i(\hat{\boldsymbol{X}}_i^b)E_m(\hat{\boldsymbol{X}}_i^c) - E_m(\hat{\boldsymbol{X}}_i^b)E_i(\hat{\boldsymbol{X}}_i^c)}{E_i(\hat{\boldsymbol{X}}_i^b) - E_i(\hat{\boldsymbol{X}}_i^c)}.$$

Due to the battery energy limitations of the slave and master nodes, $E_i = E_i(\hat{\boldsymbol{X}}_i^a)$ when $E_i < E_i(\hat{\boldsymbol{X}}_i^a)$, and $E_i = E_i(\hat{\boldsymbol{X}}_i^c)$ when $E_i > E_i(\hat{\boldsymbol{X}}_i^c)$, respectively.

Note that this off-line preparation does not need to re-operate no matter the changes of the network or the sensor nodes, except the application changes.

DOOTA

Based on the analysis in Section 4.5.1 and the results obtained by the off-line preparation, the DOOTA algorithm is presented in this part. It is actually an on-line negotiation among the slave and master nodes considering the parameters of the network, after each slave node stores the function of the *optimal curve* off-line. To help easily understand the proposed algorithm, a naive method is firstly presented, which requires a large quantity of intra-cluster communication overhead. Then, the lightweight DOOTA algorithm is further proposed.

The procedure of the naive method is described as follows. Naively, the master node firstly broadcasts an expected network lifetime T_e when the network starts to work. Each slave node computes its own expected energy consumption $E_{e\_i} = B_i/T_e$ and the corresponding cost of the master node E_{e_mi} according to (4.39), and then transmits E_{e_i} and E_{e_mi} to the master node. After receiving the messages from all slave nodes, the master node examines whether its battery energy is enough to last T_e time. If it still has residual battery energy, i.e., $B_m > T_e \sum_{i=1}^{n} E_{e\_mi}$, it broadcasts a larger T_e, otherwise a smaller one. The slave nodes calculate the corresponding E_{e_i} and E_{e_mi} again until $B_m = T_e \sum_{i=1}^{n} E_{e\_mi}$. At last, the master node broadcasts one confirm message, and the slave nodes individually calculate their own partition solutions based on the final E_{e_i}. Although this naive method is simple, the difference between the final expected network lifetime and the optimal value depends on the granularity between the two adjacent network lifetimes. The small granularity brings little difference between the final expected network lifetime and the optimal value while a large quantity of message exchanges among the slave and master nodes is needed. Therefore, too much communication overhead is produced.

To address the overhead problem while achieve the optimal maximum network lifetime, the very lightweight DOOTA algorithm is proposed. It can dramatically reduce the number of message exchanges based on Theorem 1.

Theorem 1 *When the lifetimes of the slave and master nodes are equivalent i.e.,* $T_1 = \cdots = T_n = T_m$*, the time until the first node dies is maximized.*

Proof Assuming $T_1 = \cdots = T_n = T_m = T^*$, the average energy consumption of slave node i, E_i^*, and the master node, E_m^*, over T^* DAG execution rounds are:

$$
\begin{aligned}
E_i^* &= \frac{B_i}{T^*} \\
E_m^* &= \frac{B_m}{T^*} = \sum_{i=1}^{n} E_{mi}^*
\end{aligned}
$$

When the lifetimes of the slave and master nodes do not equal each other, e.g., $T_i = T^* + \varepsilon > T^*$, where ε is an arbitrary positive real number, the following inequality is satisfied:

$$E_i = \frac{B_i}{T^* + \varepsilon} < \frac{B_i}{T^*} = E_i^*$$

According to the monotone decreasing property of Eq. (4.39), there exits:

$$E_{mi} > E_{mi}^*$$

Correspondingly, the relation between E_m and E_m^* meets:

$$E_m = \sum_{i=1}^{n} E_{mi} > E_{mi}^* = \sum_{i=1}^{n} E_{mi}^*$$

Thus, it can be obtained that:

$$T_m = \frac{B_m}{E_m} < \frac{B_m}{E_m^*} = T^*$$

According the definition of the network lifetime, $T_m = min\{T_{s1}, \cdots, T_{sn}, T_m\}$ is the network lifetime which is smaller than T^*.

The proof is similar when $T_i = T^* - \varepsilon < T^*$, in which the lifetime, $min\{T_{s1}, \cdots, T_{sn}, T_m\} = T_i$, is smaller than T^* too. ■

Unlike the naive method where the master node randomly adjusts T_e according to the received messages, the proposed DOOTA algorithm enables the master node to calculate the temporary optimal lifetime $\hat{T}$ to reduce the communication overhead. After each slave node receives the expected network lifetime T_e from the master node, it calculates not only the expected energy cost $E_{e\_i} = B_i/T_e$ and the corresponding energy cost the master E_{e_mi} as in the naive method but also the slope of the segment which passes the energy point (E_{e_i}, E_{e_mi}), k_{e_i}, as shown in Fig. 4.5.

Let $(\hat{E}_i, \hat{E}_{mi})$ represent the energy point that corresponds to the optimal network lifetime $\hat{T}$. Assuming it is still on the current segment, $\hat{E}_{mi}$ can be calculated by:

$$\hat{E}_{mi} = E_{e\_mi} + k_{e\_i}(\hat{E}_i - E_{e\_i}) \tag{4.40}$$

According to Theorem 1, there exists:

$$\hat{T} = \frac{B_i}{\hat{E}_i} = \frac{B_m}{\sum_{i=1}^{n} \hat{E}_{mi}} \tag{4.41}$$

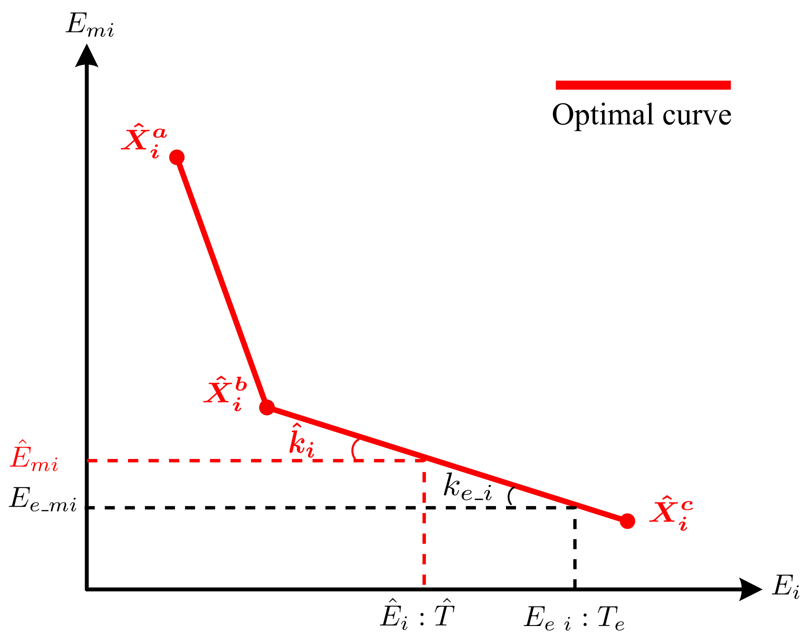


Figure 4.5: Calculation of the temporary optimal network lifetime based on the *important partition cuts*.

Since $E_{e\_i} = B_i/T_e$, combining Eqs. (4.40) and (4.41), $\hat{T}$ can be calculated in the master node by:

$$\hat{T} = \frac{B_m - T_e \sum_{i=1}^{n} k_{e\_i} E_{e\_i}}{\sum_{i=1}^{n} E_{e\_mi} - k_{e\_i} E_{e\_i}} \tag{4.42}$$

Instead of comparing B_m and $T_e \sum_{i=1}^{n} E_{e\_mi}$ in the naive method, the master node compares $\hat{T}$ with T_e in DOOTA algorithm. If they are different, it broadcasts the current expected lifetime $T_e = \hat{T}$. Slave node i repeats the calculation of E_{e_i}, E_{e_mi} and k_{e_i}, and sends them to the master node. Once $\hat{T}$ equals T_e, the master node broadcasts a $confirm$ message. The last received T_e is actually the final maximum network lifetime, based on which slave node i can easily calculate its own optimal partition cuts and the weights. For instance, assuming $\hat{T}$ in Fig. 4.5 is the final network lifetime, the optimal partition solution for slave node i consists of two partition cuts, $\hat{\boldsymbol{X}}_i^b$ and $\hat{\boldsymbol{X}}_i^c$, with the related weights p_b and p_c, can be calculated by:

$$p_b = \frac{E_i(\hat{\boldsymbol{X}}_i^c) - \hat{E}_i}{E_i(\hat{\boldsymbol{X}}_i^c) - E_i(\hat{\boldsymbol{X}}_i^b)} \tag{4.43}$$

$$p_c = \frac{E_i(\hat{\boldsymbol{X}}_i^b) - \hat{E}_i}{E_i(\hat{\boldsymbol{X}}_i^b) - E_i(\hat{\boldsymbol{X}}_i^c)} \tag{4.44}$$

The pseudo codes that executed in the master node and slave node i are shown in Algorithm 2 and Algorithm 3.

Algorithm 2 Master node algorithm

```
1: Initialize $T_e$ and broadcast it
2: for each calculation round do
3:     Receive $E_{e\_mi}$, $k_{e\_i}$ and $E_{e\_i}$
4:     Calculate $\hat{T}$ using Eq. (4.42)
5:     if $\hat{T} == T_e$ then
6:         Broadcast $confirm$
7:         Break
8:     else
9:         $T_e = \hat{T}$, and broadcast $T_e$
10:    end if
11: end for
```

Algorithm 3 Slave node algorithm

```
1: for each received message do
2:     if not $confirm$ then
3:         Calculate $E_{e\_i}$, $E_{e\_mi}$ and $k_{e\_i}$, using Eq. (4.39),
4:         and transmit them
5:     else
6:         $\hat{E}_i = B_i/T_e$
7:         Calculate partition weights using Eqs. (4.43) and (4.44)
8:         Break
9:     end if
10: end for
```

4.6 Simulation Results

In this section, extensive simulations are employed to estimate the performance of the proposed task allocation algorithms, CSTA, CDTA and DOOTA for the symmetric and asymmetric networks, respectively. Since each cluster in the network can work independently, the proposed algorithms are carried out within the cluster in this chapter. The comparisons with the previous studies for general cluster based WSNs, as well as the performances when using the real applications and the physical implementations, will be demonstrated in Chapter 5.

4.6.1 Simulation Setup

The values of the energy related parameters for calculating the energy consumption of the slave and master nodes in the networks are obtained from CC2538 system-on-chip datasheet [96] as shown in Table 4.1. The slave and master nodes are considered to use the same RF module which works at the $2.4GHz$ ISM band with a bandwidth of 250 $Kbps$.

Table 4.1: Values of the energy related parameters for calculating the energy consumption of the slave and master nodes in the networks based on CC2538 system-on-chip

Processing energy cost parameters	f_i, MCU processing speed of slave node i	32 MHz
	f_m, MCU processing speed of the master node	32 MHz
	P_i, processing power of slave node i	36.9 mW
	P_m, processing power of the master node	36.9 mW
Communication energy cost parameters	e_o, communication overhead energy cost	3.69 uJ
	t_{rx}, the time of RF for receiving 1 bit of data packet	4 μs
	t_{tx}, the time of RF for transmitting 1 bit of data packet	4 μs
	P_{T0}, energy cost of electronic circuits of the RF for receiving or transmitting 1 bit of data packet	59.8 mW
	P_{rin}, receiver sensitivity	-85 dBm
	η, drain efficiency	0.05
	F, RF frequency	2.4 GHz
	α, the path loss exponent	2

The DAG graphs for the applications are randomly generated based on the number of vertexes and edges. The computation workload of the vertexes in the DAG graphs are distributed within the range of $[100, 1000]$ kilo clock cycles (KCC). The amount of the communicated data on the edges are distributed in the range of $[100, 500]$ bits.

4.6.2 Evaluations of CSTA, CDTA and DOOTA for Symmetric WSNs

The cluster is generated with 1 master node and n slave nodes. Like the common assumption for the symmetric networks as in [29], this section considers that all the slave nodes have the same battery energy and transmitting power. The network lifetime increase by using the proposed task allocation algorithms with respect to the *no-scheduling* scheme, in which the slave node only executes the first task and the rest tasks are done by the master node, as well as the execution time of running them in Matlab 2017a are investigated by changing the following parameters:

- The number of slave nodes, n;
- The ratio of the master node's battery energy to the slave nodes', R_{ms};
- The number of tasks of the application, K;
- The variation among the tasks.

The configuration parameters are summarized in Table 4.2, and only one parameter is changed in each experiment. Among the configuration parameters, the definitions of different variation levels are described as:

- Low variation level: the workload of the tasks are randomly generated;
- Middle variation level: Based on the low level, randomly select one task and enlarge its workload 10 times;
- High variation level: Based on the low level, randomly select one task and enlarge its workload 50 times.

The reported results correspond to the **average values** and the **standard deviations** based on running 500 test instances for each simulation.

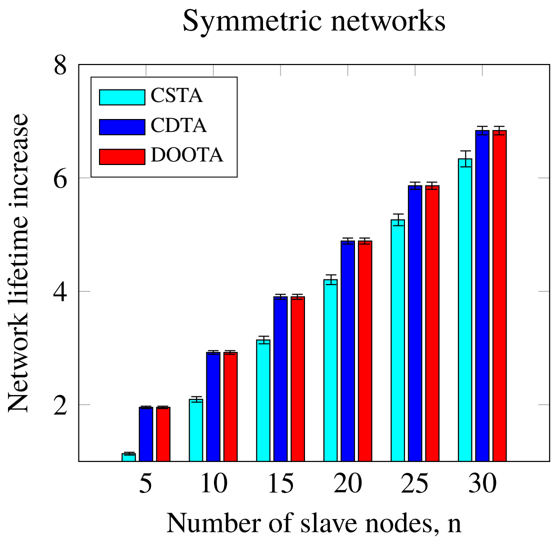
Table 4.2: Configuration parameters of the simulations for symmetric networks.

Parameters	Values	
	Default	Varied
Number of slave nodes, n	10	{5, 10, 15, 20, 25, 30}
Ratio of the battery energy of the master node to the slave node, R_{ms}	5	{ 0.5, 1, 5, 10}
Number of tasks of the application, K	10	{5, 10, 15, 20, 25}
Variation level among the tasks	middle level	{ low, middle, high}

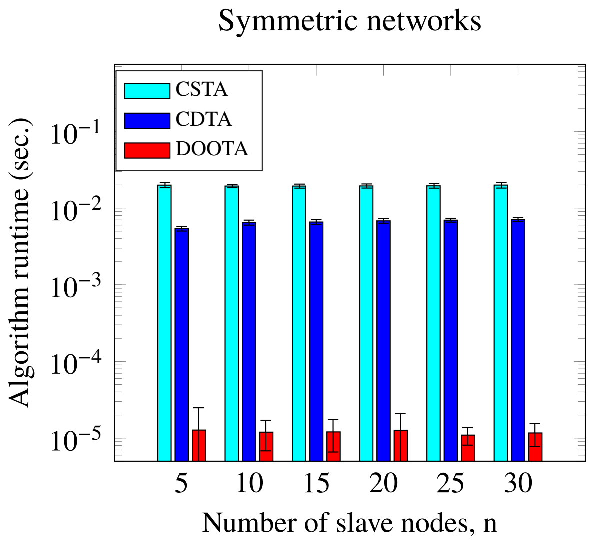
Effect of the number of the slave nodes for symmetric WSNs

The first set of simulations is conducted in this part to estimate the performance of the proposed algorithms for symmetric networks when changing the number of the slave nodes.

Fig. 4.6 depicts the network lifetime increase by using the proposed task allocation algorithms with respect to the no-scheduling strategy and the corresponding algorithm runtime. It can be easily obtained that the network lifetime increases by using the three proposed algorithms become more significant when the number of slave nodes n changes from 5 to 30. Taking the DOOTA algorithm for example, it extends the network lifetime in average from 1.95 to 6.84 times longer than the no-scheduling strategy. This can be explained by the fact that the master node exponentially gets overloaded using the no-scheduling strategy as n increased, which leads it die soon. While the workload of all the slave and master nodes can be efficiently allocated by the task allocation algorithms, which makes the network stay active for longer time. Among these three task allocation algorithms, CSTA improves the network lifetime the least, due to the fact that it only provides the static task allocation solution. Rather than using the static solution, both CDTA and DOOTA extend the network lifetime longer by unitizing multiple partition cuts with the corresponding weights. Specially, they achieve the same network lifetime increase since they provide exactly the same task allocation solutions. This remarkable phenomenon validates the analysis of the optimal task allocation solution in Section 4.5.1.



(a)



(b)

Figure 4.6: Effect of the number of slave nodes in the cluster on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms in symmetric networks, respectively (The ratio of the battery energy of the master node to the slave node is $R_{ms} = 5.0$, there are $K = 10$ tasks in the application and the middle variation level is selected, respectively).

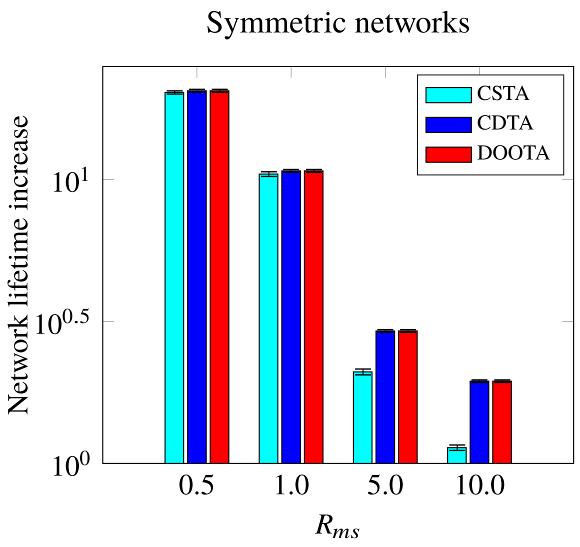
Moreover, the superiority of CDTA and DOOTA over CSTA on extending the network lifetime decreases when n becomes larger as Fig. 4.6a shows. Specifically, CDTA and DOOTA extend the network lifetime in average 171.92% longer than CSTA when n equals 5, while their gains decrease to 107.87% when n equals 30. Since the master node has to be in charge of all the slave nodes, the partition cut that enables the master node consume the least energy will be chosen as the only solution by both the static and dynamic task allocation approaches for a large number of slave nodes.

Fig. 4.6b compares the execution time of running the three task allocation algorithms. It is obvious that executing both the centralized algorithms CSTA and CDTA requires plenty of time than DOOTA. The algorithm runtime of DOOTA is 3 orders of magnitude smaller. Besides, the execution time of these three algorithms remain stable as n changes. The reason is that the slave nodes are considered as the same in the symmetric networks and they use the same partition solution.

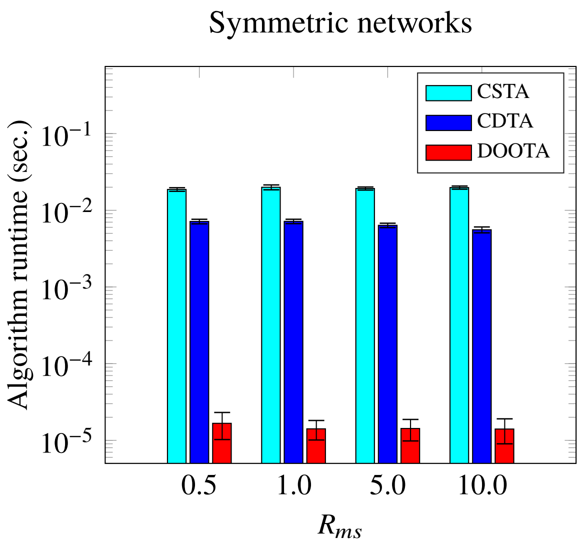
Effect of the ratio of the battery energy of the master to the slaves for symmetric WSNs

In the second set of simulations, ratio of the battery energy of the master node to the slave node, R_{ms}, is varied to investigate the impact on the algorithms' performance with different battery energy.

Fig. 4.7a shows that the network lifetime increase by applying the task allocation algorithms with respect to the no-scheduling strategy is very significant when R_{ms} is small. For instance, an average gain of 2052.91% is achieved by DOOTA when R_{ms} is 0.5. When the master node has more battery energy, i.e., R_{ms} becomes larger, the gains decreases. The major reason is that the master node dies very fast by using no-scheduling strategy when it has a small capacity of battery. While the task allocation algorithms efficiently prolong the network lifetime by balancing the energy cost of the slave and master nodes. When the battery energy of the master node is very large, the partition solutions obtained by the task allocation algorithms are very close to the no-scheduling strategy. Therefore, the performance of the proposed algorithms on



(a)



(b)

Figure 4.7: Effect of the ratio of the master node's battery energy to the slave node on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms in symmetric networks, respectively (There are $n = 10$ slave node in the cluster and $K = 10$ tasks in the application, and the middle variation level is selected, respectively).

extending the network lifetime is more significant for small R_{ms}.

Furthermore, DOOTA performs as well as CDTA, which is consistent with the simulation results in Fig. 4.6. The superiority of both the DOOTA and CDTA on extending the network lifetime over the CSTA increases as R_{ms} becomes larger. The network lifetime is extended almost the same by these three algorithms when R_{ms} equals 0.5, while DOOTA and CDTA extend the network lifetime in average 171.51% longer than CSTA when R_{ms} equals 10. If the master node has a small capacity of the battery, the partition solutions obtained by these three algorithms are very similar, and they tend to be very close to or even the same as the one which enables the master node consume the least energy. When the battery energy of the master node is large, there are more possibilities for DOOTA and CDTA to select different partition cuts to achieve more balanced workload distribution among the slave and master nodes.

The execution time of running the algorithms are presented in Fig. 4.7b. CSTA, CDTA and DOOTA require constant time to run the algorithms, due to the same reason as presented in the above section.

Effect of the number of the tasks in the application for symmetric WSNs

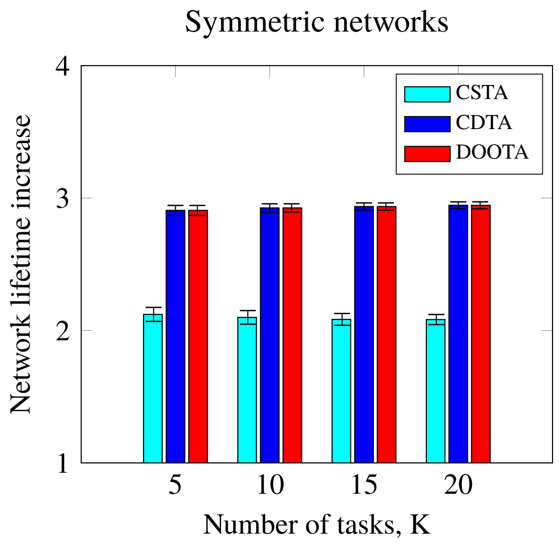
In addition to the impact factors from the nodes and networks, the application effect on the performance of the proposed algorithms is further investigated. Firstly, the effect of the number of the tasks in the application K is evaluated in this part.

It can be seen from Fig. 4.8a that the network lifetime increase by using the proposed algorithms with respect to no-scheduling strategy slightly vary as K changes from 5 to 20. This can be explained as follows. On the one hand, the workload assigned to the master node by the no-scheduling strategy increases as K becomes larger. On the other hand, although the task allocation algorithms can efficiently balance the workload among the slave and master nodes, the master node still undertakes more workload due to the increasing tasks.

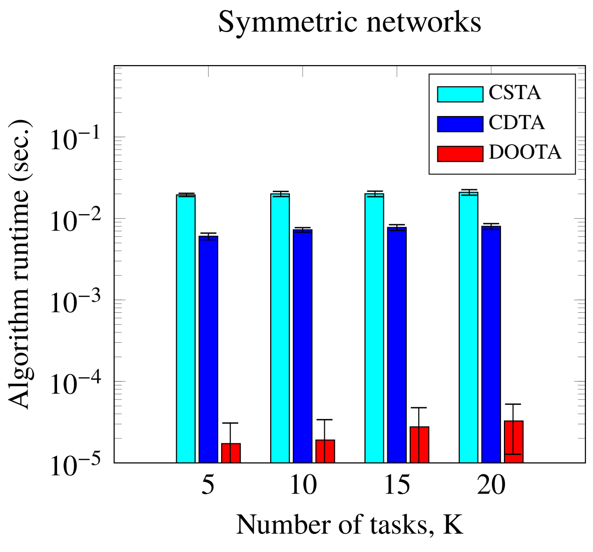
Fig. 4.8b illustrates the execution time of running the algorithms. The algorithm runtime of CSTA and CDTA remain stable as K changes from 5 to 20, while the algorithm runtime of DOOTA slightly increases. Since DOOTA is based on the important partition cuts, the number of which may increase as K becomes large. However, the algorithm runtime of DOOTA is still 3 orders of magnitude smaller than the centralized algorithms CSTA and CDTA.

Effect of the variation among the tasks for symmetric WSNs

This part further investigates the performance of the task allocation algorithms by changing the variation levels among the tasks in the applications. As depicted in Table 4.2, three variation levels are selected in this set of simulations.



(a)

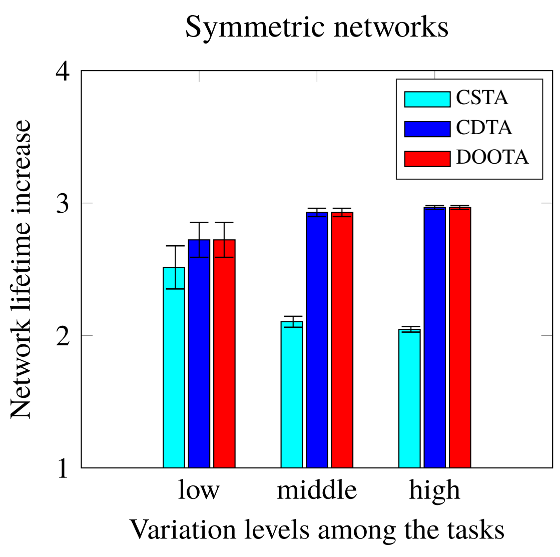


(b)

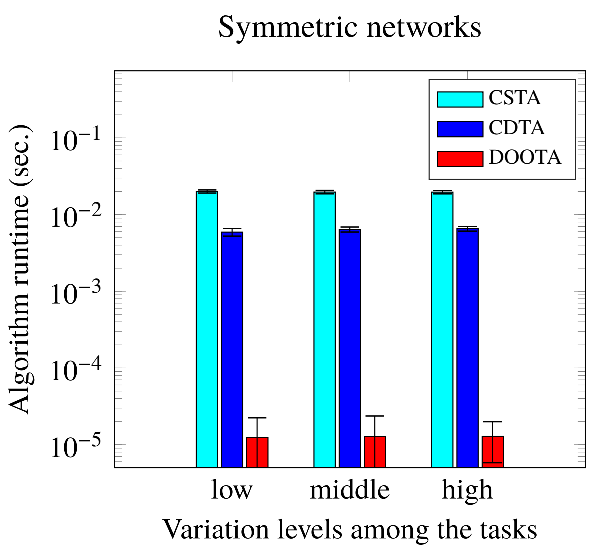
Figure 4.8: Effect of the number of tasks in the application on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms in symmetric networks, respectively (There are $n = 10$ slave node in the cluster, the ratio of the battery energy of the master node to the slave node is $R_{ms} = 5.0$, and the middle variation level is selected, respectively).

As shown in Fig. 4.9a, the trends of the network lifetime in average increase by using CDTA and DOOTA task allocation algorithms slightly go up. Both CDTA and DOOTA improve the network lifetime in average from 272.21% to 296.64% as the variation level changes from low to high. On the contrary, the gain of the network lifetime increase by using CSTA decreases from 251.44% to 204.65%. According to the definitions of the variation levels, the workload of the whole tasks increases for a higher variation level. DOOTA and CDTA efficiently enable the slave and master nodes fairly share the heavy tasks by using multiple partition cuts. While CSTA only provides static partition solution and suffers a lot from the heavy tasks. Hence, the dynamic task allocation algorithms using multiple partition solutions are more suitable for the applications with high variation among the tasks.

The execution time of the proposed algorithms do not vary as the variation level among the tasks changes as depicted in Fig. 4.9b. Since the different variation levels do not change the number of the tasks as well as the number of the important partition cuts, the time requirements for running CSTA, CDTA and DOOTA are not affected.



(a)



(b)

Figure 4.9: Effect of the variation among the tasks on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms in symmetric networks, respectively (There are $n = 10$ slave node in the cluster and $K = 10$ tasks in the application, and the ratio of the battery energy of the master node to the slave node is $R_{ms} = 5.0$, respectively).

4.6.3 Evaluations of CSTA, CDTA and DOOTA for Asymmetric WSNs

Compared with the symmetric networks, asymmetric networks are more commonly used in the real scenarios. Therefore, this section investigates the performance of the proposed algorithms for asymmetric networks. To be close to the realistic scenarios, the cluster is randomly generated in a two dimension area of 100×100 square meters with one master node and n slave nodes. The master node is located at the center and the n slave nodes are randomly distributed in the area. Since the transmitting power is a function of the distance as presented in Section 3.3.1, the slave nodes may have different transmitting powers due to the randomly generated positions. Besides, the battery energy of the master node and the slave nodes are randomly generated within the ranges $[6kJ, 10kJ]$ and $[1kJ, 5kJ]$, respectively.

Like the evaluations for symmetric networks, the performance of the proposed algorithms on network lifetime increase with respect to the no-scheduling strategy and the algorithm runtime are investigated by changing the number of slave nodes, n, the number of the tasks in the application, K, and the variation level among the tasks. The same configuration parameters are used as listed in Table 4.2 except the ratio of the battery energy of the master node to the slave node, due to the fact that the battery energy of the slave and master nodes are randomly generated for the asymmetric networks. 500 test instances are run for each simulation and the reported results correspond to the average values and the standard deviations.

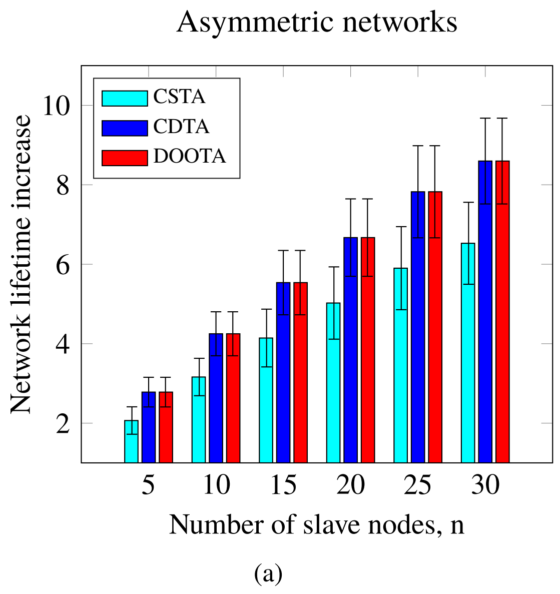
Effect of the number of the slave nodes for asymmetric WSNs

In this part, a set of simulations are conducted to estimate the performance of the proposed algorithms for asymmetric networks when changing the number of the slave nodes.

The network lifetime increase by using the proposed task allocation algorithms with respect to the no-scheduling strategy and the corresponding algorithm runtime are depicted in Fig. 4.10. Like the results in the symmetric networks, the network lifetime increases by using the three proposed algorithms become more significant when the number of slave nodes n increases, e.g., DOOTA extends the network lifetime in average 278.15% longer than the no-scheduling strategy when n equals 5 and this gain increases to 859.78% when $n = 30$. The reason is the same as presented in symmetric networks that the master node quickly gets overloaded using the no-scheduling strategy as n increased, which leads it die soon. While the workload of all the slave and master nodes can be efficiently allocated by the task allocation algorithms, which makes the network stay active for longer time. Due to the fact that CSTA only provides the static partition solution while CDTA and DOOTA supply multiple partition solutions, CSTA performs the worst among these three algorithms on extending the network lifetime. Besides, DOOTA improves the network lifetime as long as CDTA, which again validates the analysis of the optimal task allocation solution in Section 4.5.1.

Unlike the results in symmetric networks shown in Fig. 4.6a, the superiority of CDTA and DOOTA over CSTA on extending the network lifetime increases when n changes from 5 to 30 for the asymmetric networks. Fig. 4.10a shows that CDTA and DOOTA extend the network lifetime in average 71.53% longer than CSTA when n equals 5 with respect to no-scheduling strategy, while their gains increase to 206.95% when n equals 30. Each slave node may have different partition solutions in the asymmetric networks due to different positions and battery energy. The partition solutions for each slave node provided by CDTA and DOOTA, which consist of multiple partition cuts and the corresponding weights, are better than the static partition cuts provided by CSTA. Therefore, when n increases, the benefits of using CDTA and DOOTA accumulates.

Since it needs to consider the partition solutions for each slave nodes in the asymmetric networks, the execution time of running CSTA, CDTA and DOOTA increases as n becomes larger. Taking CSTA for example, it takes in average 7.93×10^{-2} seconds when there are 5 slave nodes, while 5.98×10^{-1} seconds are needed as n changes to 30. Although the trend of the algorithm runtime of DOOTA increases like the other two centralized algorithms, it is still 3 and 2 orders



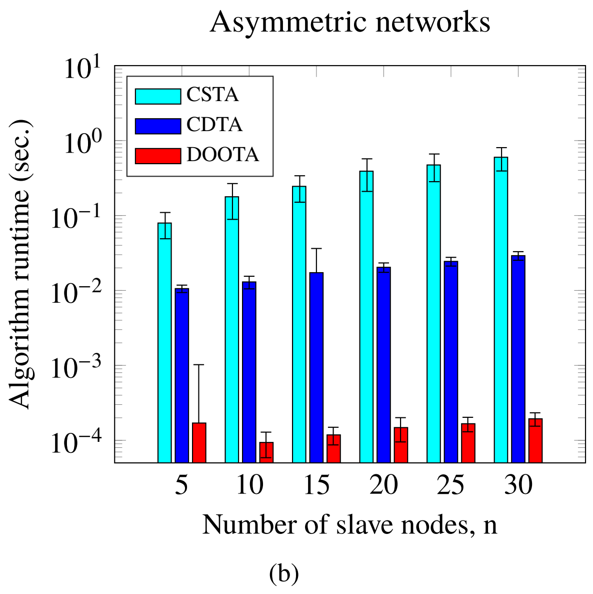


Figure 4.10: Effect of the number of slave nodes in the cluster on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms, respectively (There are $K = 10$ tasks in the application and the middle variation level is selected, respectively).

of magnitude smaller than CSTA and CDTA.

Effect of the number of the tasks in the application for asymmetric WSNs

The second set of simulations are conducted to investigate the effect of the number of tasks in the application, K, on the performance of the proposed algorithms for the asymmetric networks.

Fig. 4.11a shows the network lifetime increase by using the task allocation algorithms for asymmetric networks. The gains of CSTA, CDTA and DOOTA with respect to the no-scheduling strategy are slightly changed as K varies. This phenomenon is consistent with the results for symmetric networks, which can be explained by the same reasons.

The execution time of running CSTA, CDTA and DOOTA task allocation algorithms for asymmetric networks are illustrated in Fig. 4.11b. The centralized algorithms, CSTA and CDTA, require drastically more execution time when K becomes larger. Specifically, CSTA takes 4.99×10^{-2} seconds in average as K equals 5 while it needs almost 1 second when there are 20 tasks. Since the partition cut is modeled as a $K \times 1$ binary vector as presented in Section 4.3.2, K determines the complexities of the centralized algorithms. For asymmetric networks, in which the slave nodes need individual partition solutions, the complexities of calculating the partition solutions are exponentially increases. In contrast to the centralized algorithms, the execution time of running DOOTA changes less. The reason is that DOOTA calculates the optimal partition solutions based on the number of the important partition cuts. Although the number of the tasks in the application increase, the important partition cuts may not change or slightly changes. Further more, the algorithm runtime of DOOTA is still 3 and 2 orders of magnitude smaller than the centralized algorithms CSTA and CDTA.

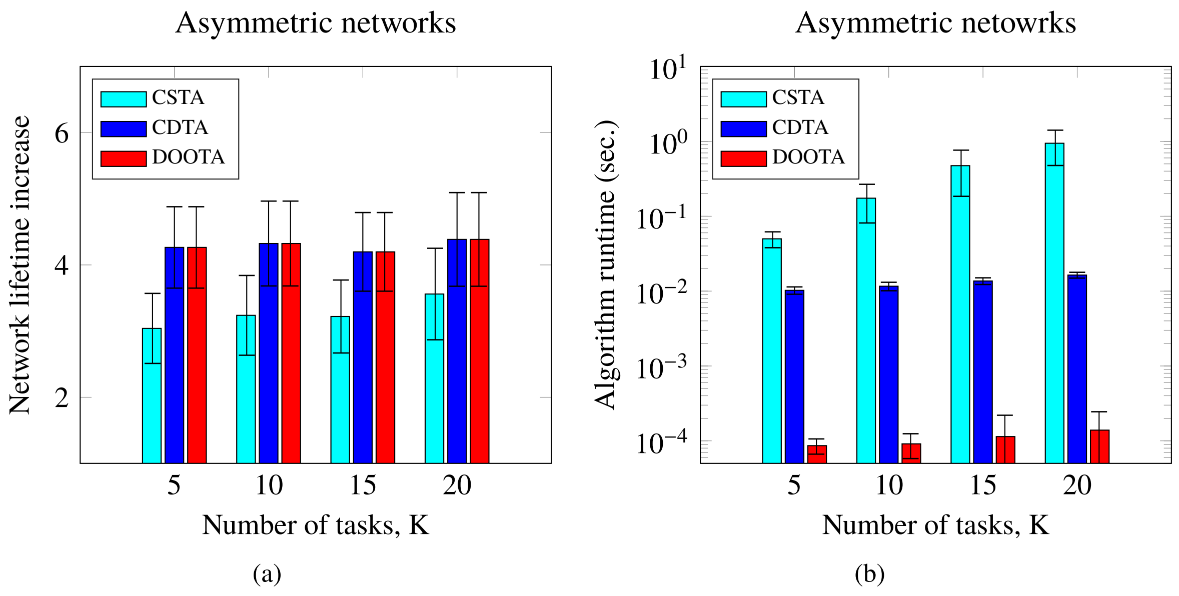
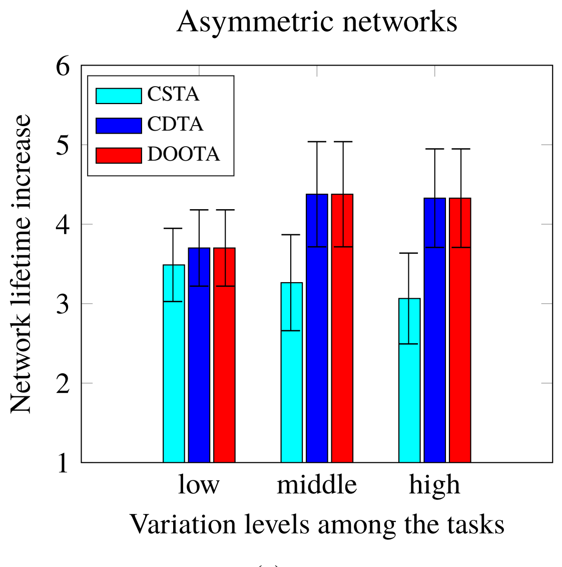


Figure 4.11: Effect of the number of tasks in the application on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms, respectively (There are $n = 10$ slave node in the cluster and the middle variation level is selected, respectively).

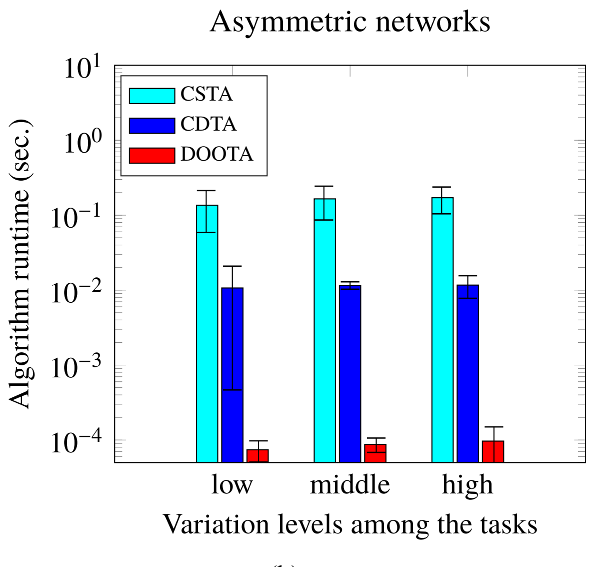
Effect of the variation among the tasks for asymmetric WSNs

The performance of the task allocation algorithms are further investigated by changing the variation levels among the tasks in the applications. The same three variation levels, low variation level, middle variation level and high variation level, are used in this set of simulations.

The trend of the network lifetime increase by using CSTA, CDTA and DOOTA task allocation algorithms are very similar to the results for the symmetric networks. Fig. 4.12a illustrates that both the CDTA and DOOTA algorithms improve the network lifetime longer as the variation levels raise, i.e., they extend the network lifetime longer from 370.05% to 437.72% in average when it changes from the low variation level to the high variation level. On the contrary, the performance of CSTA becomes weaker where the variation level is higher. Since CSTA only provides the static partition cut, which cannot achieve a fair workload balance among the slave and master nodes for the tasks with high variations. This drawback of the static task allocation can be solved by the dynamic task scheduling using multiple partition cuts. Thus, DOOTA and CDTA performs much better than CSTA.



(a)



(b)

Figure 4.12: Effect of the variation among the tasks on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms, respectively (There are $n = 10$ slave node in the cluster and $K = 10$ tasks in the application, respectively).

Fig. 4.12b depicts the corresponding executing time of running these three task allocation algorithms. Since the algorithms need to consider the partition solution for each slave node, their algorithm running time are larger by comparing with running them for symmetric networks. However, the variation levels do not change the number of the tasks as well as the number of the important partition cuts, the time requirements for running CSTA, CDTA and DOOTA remain stable.

4.7 Summary

Energy efficiency is a primary concern in almost every WSN application. This chapter has presented the energy aware task allocation algorithms to achieve the fair balanced energy cost across the network and to maximize the network lifetime. Firstly, system models including the network structure, the application models and the cost functions of the sensor nodes are presented. The cluster based hierarchical WSNs are selected, and the WSN applications are modeled by the DAG graphs. Meanwhile, the systemic energy cost of the WSN nodes are formulated by taking the computation and communication cost into account. Based on these system models, this chapter then describes three task allocation algorithms, CSTA, CDTA and DOOTA.

The problem of distributing the tasks of the applications is formulated as partitioning the modeled DAG graph into two subgraphs: one for the slave node and the other is executed by the master node. CSTA formulates the workload distribution problem as a BILP problem by introducing one binary vector variable to represent the partition cuts. This algorithm is executed by the gateway, and it provides the static partition solutions which enables each slave node apply one fixed time invariant partition cut to balance the workload distribution of the tasks. Furthermore, CSTA is extended to CDTA by using multiple partition cuts with different weights to achieve more balanced workload distribution among the slave and master nodes. By using the mean energy cost of the slave and master nodes at each DAG execution round, CDTA formulates the dynamic task allocation problem as a LP problem. However, CDTA is a centralized algorithm as well as CSTA, it also suffers from the complex computation. Meanwhile, the centralized algorithms require to collect all of the parameters of the nodes, networks and the applications in advance. These drawbacks make CSTA and CDTA are hard to achieve online and incur large delays. To overcome the drawbacks of the centralized algorithms, DOOTA, a distributed optimal on-line task allocation algorithm, is further proposed in this chapter. It firstly presents a indepth analysis to prove that the optimal partition solutions consist of at most two partition cuts. Based on the extracted important partition cuts, DOOTA enables the slave and master negotiate on-line to calculate the optimal partition solutions.

Extensive simulations are conducted to investigate the performance of CSTA, CDTA and DOOTA in both symmetric and asymmetric networks. The results demonstrate that the three algorithms can drastically extend the network lifetime. Especially, the network lifetime increase is more significant when the cluster contains large number of slave nodes. Taking the asymmetric networks with 30 slave nodes for example, CSTA, CDTA and DOOTA extend the network lifetime in average 6.53, 8.60 and 8.60 times longer with respect to the no-scheduling strategy, respectively. Moreover, as expected, both CDTA and DOOTA extend the network lifetime longer than CSTA, since CSTA only provides the static partition solution. Their superiorities are more obvious for the tasks with high variation levels. Further on, DOOTA performs as well as CDTA on extending the network lifetime, which also validates the optimality of DOOTA. Besides, the time requirement for running the distributed on-line algorithm DOOTA is up to 3 and 2 orders of magnitude smaller than the centralized algorithms CSTA and CDTA. This characteristic indicates that DOOTA is a very lightweight algorithm, which provides a strong advantage for the on-line calculation.

5 Extensions for Different Task Scenarios and Network Structures

5.1 Introduction **71**

5.2 Extensions of Task Allocation for Different Task Scenarios **73**

5.2.1 Task Allocation Algorithms for Condition Triggered Tasks: SCTTA and DCTTA 73

5.2.2 Task Allocation Algorithms for Joint Local and Global Tasks: SJTA and DJTA 79

5.3 Extensions of Task Allocation for Multi-hop Mesh WSNs: DTA-mhop . . **86**

5.3.1 Modeling the Task Allocation for Multi-hop Mesh WSNs 86

5.3.2 DTA-mhop Algorithm 87

5.4 Simulation Results **92**

5.4.1 Simulation Setup 92

5.4.2 Evaluation of SCTTA and DCTTA Algorithms 93

5.4.3 Evaluation of SJTA and DJTA Algorithms 97

5.4.4 Evaluation of DTA-mhop Algorithm 100

5.5 Summary **104**

5.1 Introduction

During the past decades, wireless sensor networks WSNs have been applied to a wide variety of applications with vastly varying requirements and characteristics. Although centralized and distributed task allocation algorithms have been proposed in Chapter 4 for extending the lifetimes of cluster based WSNs, they are not sufficient for all of the various WSN applications. This chapter extends the proposed task allocation algorithms for different task scenarios and network structures.

Specifically, the possibilities of using two different task scenarios and the multi-hop mesh network structure are considered in this chapter. Section 5.2 presents the extensions of the task allocation algorithms for different task scenarios for cluster based WSNs, i.e., for condition triggered tasks and the joint local and global tasks. In the scenario of conditional tasks, such as applications in event driven WSNs, the slave nodes are required to execute different tasks according to the conditions. The condition triggered application can be modeled by a directed acyclic graph (DAG) with conditional branches. This conditional DAG graph is further decomposed into multiple standard DAG (static DAG without conditional branches) with different weights according to the satisfaction probability of each condition. Based on this model, a static and a dynamic condition triggered task allocation algorithms (SCTTA and DCTTA) are proposed based on CSTA and CDTA as presented in Chapter 4 by considering the decomposed DAG graphs simultaneously.

In the scenario of joint tasks, the slave and master nodes are required to collaborate to accomplish the global tasks periodically in addition to the local tasks. For example, in model based compression approaches, such as [24, 97], the system models need to be updated based on the received data from all of the slave nodes over a period of time. The local task allocation problem is modeled as in the previous chapter, while the global task allocation problem is modeled by dividing the global DAG graph into multiple subgraphs mapping to the slave and master nodes. Then, a static and a dynamic joint task allocation algorithms (SJTA and DJTA) are proposed by formulating the joint task allocation problem as a series of binary integer linear programming (BILP) and linear programming (LP) problems.

Furthermore, since the multi-hop mesh network structure is also very popular compared with the cluster based hierarchical networks, Section 5.3 illustrates the task allocation algorithm for multi-hop mesh networks (DTA-mhop). By considering that the routing nodes have the abilities to share the tasks of the neighbor nodes, maximizing the lifetime of the multi-hop networks is formulated by a LP problem.

Finally, Section 5.4 estimates the proposed task allocation algorithms by extensive simulation results, and Section 5.5 summarizes this chapter.

5.2 Extensions of Task Allocation for Different Task Scenarios

This section presents the task allocation algorithms for condition triggered applications and the joint local and global applications in Section 5.2.1 and Section 5.2.2, respectively. They are both developed based on the centralized task allocation algorithms proposed in Chapter 4 for cluster based hierarchical WSNs. For generality, the asymmetric networks are considered in which the positions and the battery energy of the slave and master nodes vary. Note that, the proposed task allocation algorithms are still carried out within the cluster, since each cluster can work independently [42].

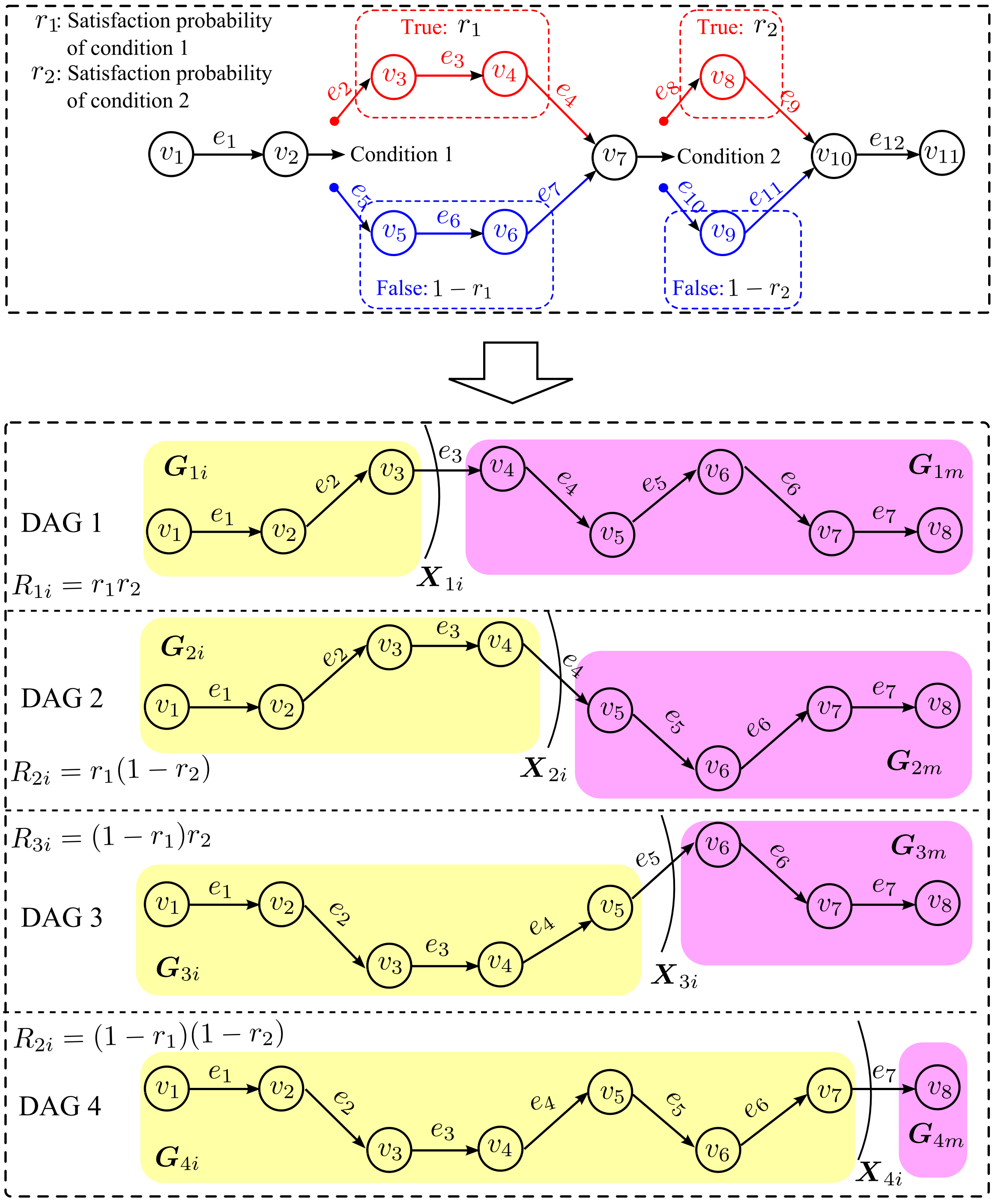
5.2.1 Task Allocation Algorithms for Condition Triggered Tasks: SCTTA and DCTTA

The mechanism of condition triggers has been widely used in cluster based WSNs to conserve the energy consumption of slave nodes. This section extends the previous proposed task allocation algorithms and proposes a static condition triggered task allocation algorithm (SCTTA) and a dynamic condition triggered task allocation algorithm (DCTTA), respectively.

Modeling the Condition Triggered Task Allocation

The condition triggered WSN application may have multiple conditions, and each condition is considered to have two possibilities: either satisfied (true) or not (false). According to this characteristic, given a specific condition triggered WSN application, it can be modeled by a DAG graph with conditional branches. In order to model the condition triggered task allocation problem, the modeled conditional DAG graph is further decomposed into multiple stationary DAG graphs without conditional branches according to the satisfaction probability of each condition. Then, by using the idea of CSTA algorithm, the problem of task allocation for conditional tasks is equivalent to partition each of the divided DAG graph into two parts and distribute to the slave and master nodes.

Fig. 5.1 illustrates one example of the task allocation for a conditional application which has two conditions and the corresponding satisfaction probabilities are r_1 and r_2. The modeled conditional DAG graph is divided into 4 stationary DAG graphs. The probabilities of slave node i for executing the 4 stationary DAG graphs are $R_{1i} = r_1 r_2$, $R_{2i} = r_1(1 - r_2)$, $R_{3i} = (1 - r_1)r_2$ and $R_{4i} = (1 - r_1)(1 - r_2)$, which satisfy: $R_{1i} + R_{2i} + R_{3i} + R_{4i} = 1$. Consequently, the task allocation



r_1: Satisfaction probability of condition 1
r_2: Satisfaction probability of condition 2
True: r_1
True: r_2
Condition 1
Condition 2
False: $1-r_1$
False: $1-r_2$
DAG 1
$R_{1i} = r_1 r_2$
DAG 2
$R_{2i} = r_1(1-r_2)$
DAG 3
$R_{3i} = (1-r_1)r_2$
DAG 4
$R_{2i} = (1-r_1)(1-r_2)$
$\boldsymbol{G}_{1i}$, $\boldsymbol{G}_{1m}$, $\boldsymbol{X}_{1i}$
$\boldsymbol{G}_{2i}$, $\boldsymbol{G}_{2m}$, $\boldsymbol{X}_{2i}$
$\boldsymbol{G}_{3i}$, $\boldsymbol{G}_{3m}$, $\boldsymbol{X}_{3i}$
$\boldsymbol{G}_{4i}$, $\boldsymbol{G}_{4m}$, $\boldsymbol{X}_{4i}$

Figure 5.1: Schematic diagram of task allocation for condition triggered WSN application.

for this conditional DAG graph is modeled by partitioning the 4 stationary DAG graphs into two parts with 4 partition cuts, $\boldsymbol{X}_{1i}$, $\boldsymbol{X}_{2i}$, $\boldsymbol{X}_{3i}$ and $\boldsymbol{X}_{4i}$, respectively.

Observe that the amount of the transmitted data from slave node i to the master node when executing one of the divided stationary DAG graphs is the summation of the weights of the edges which cross the corresponding partition cut. Taking Fig. 5.1 for example, slave node i will transmit $l(e_3)$ bits of data when it executes DAG 1. For each stationary DAG graph, the constraints for guaranteeing a) the collaboration between the slave and master nodes and b) the direction of the transmitted data can be formulated as the same as Eqs. (4.1) and (4.2) in Section 4.3.1.

Task Allocation Algorithms for Condition Triggered Tasks

According to the above analysis, the condition triggered task allocation can be treated as allocating the tasks of multiple stationary DAG graphs simultaneously. Hence, this section proposes the SCTTA and DCTTA algorithms based on the previous static and dynamic task allocation algorithms, CSTA and CDTA, for the scenario of condition triggered tasks.

Considering a general condition triggered application, it can be firstly divided into Γ stationary DAG graphs according to different conditions. Let $R_{\gamma i}$ ($\gamma = 1, \cdots, \Gamma$) denote the probability of slave node i for executing stationary DAG γ, it can be calculated according to the satisfactory probability of the each condition. For all of the Γ stationary DAG graphs, there exists $\sum_{\gamma=1}^{\Gamma} R_{\gamma i} = 1$. The partition cut for stationary DAG γ is represented by a binary vector variable $\boldsymbol{X}_{\gamma i} = [x(v_1), \cdots, x(v_k), \cdots, x(v_{K\gamma})]^T$. $K\gamma$ is the number of vertexes in the divided DAG γ, and $x(v_k)$ equals 1 if vertex v_k belongs to slave node i and 0 otherwise. According to Eqs. (4.4) and (4.5), the constraints a) and b) are formulated as:

$$1 \leq \mathbf{1}_{1\times K\gamma}\boldsymbol{X}_{\gamma i} \leq K\gamma - 1 \tag{5.1}$$

$$\boldsymbol{B}_{\gamma}^{T}\boldsymbol{X}_{\gamma i} \geqq \mathbf{0} \tag{5.2}$$

The energy cost of slave node i for applying the partition cut $\boldsymbol{X}_{\gamma i}$ is denoted by $E_{\gamma i}(\boldsymbol{X}_{\gamma i})$. It is made up of the energy cost of executing the assigned tasks and transmitting data to the master node. Based on Eq. (4.8), $E_{\gamma i}(\boldsymbol{X}_{\gamma i})$ can be expressed as:

$$E_{\gamma i}(\boldsymbol{X}_{\gamma i}) = \boldsymbol{E}_{p\gamma\_i}\boldsymbol{X}_{\gamma i} + e_o + e_{tx}(d_i)\boldsymbol{L}_{1n}\boldsymbol{X}_{\gamma i} \tag{5.3}$$

where $\boldsymbol{E}_{p\gamma\_i} = [E_{pi}(v_1), \cdots, E_{pi}(v_{K1})]$ stands for the processing cost of the vertexes in DAG γ when they are executed by slave node i; $\boldsymbol{L}_{\gamma n} = [l_n(v_1), \cdots, l_n(v_{K\gamma})]$ represents the net generated data of the vertexes in DAG γ. Similarly, according to Eq. (4.9), the corresponding energy cost of the master node for applying the partition cuts $\boldsymbol{X}_{\gamma i}$ and is:

$$E_{m\gamma\_i}(\boldsymbol{X}_{\gamma i}) = \boldsymbol{E}_{p\gamma\_m}(\mathbf{1}_{K\gamma\times 1} - \boldsymbol{X}_{\gamma i}) + e_o + e_{rx}\boldsymbol{L}_{\gamma n}\boldsymbol{X}_{\gamma i} \tag{5.4}$$

where $\boldsymbol{E}_{p\gamma\_m} = [E_{pm}(v_1), \cdots, E_{pm}(v_{K\gamma})]$ represents the processing cost of the vertexes in DAG γ when they are executed by the master node.

As γ changes from 1 to Γ, the energy cost of slave node i and the master node for applying the partition cut $\boldsymbol{X}_{\gamma i}$ vary. In order to formulate the network lifetime, the average energy cost of the slave and master nodes, $E_i(\boldsymbol{X}_{\gamma i})$ and $E_{m\_i}(\boldsymbol{X}_{\gamma i})$, are introduced. Based on the execution probabilities of the stationary DAG graphs, $E_i(\boldsymbol{X}_{\gamma i})$ and $E_{m\_i}(\boldsymbol{X}_{\gamma i})$ can be formulated as:

$$\begin{aligned} E_i(\boldsymbol{X}_{1i,\cdots,\Gamma i}) &= \sum_{\gamma=1}^{\Gamma} R_{\gamma i}E_{\gamma i}(\boldsymbol{X}_{\gamma i}) \\ &= \sum_{\gamma=1}^{\Gamma} R_{\gamma i}\big(\boldsymbol{E}_{p\gamma\_i}\boldsymbol{X}_{\gamma i} + e_o + e_{tx}(d_i)\boldsymbol{L}_{\gamma n}\boldsymbol{X}_{\gamma i}\big) \end{aligned} \tag{5.5}$$

$$\begin{aligned} E_m(\boldsymbol{X}_{1i,\cdots,\Gamma i}) &= \sum_{\gamma=1}^{\Gamma} R_{\gamma i}E_{m\gamma\_i}(\boldsymbol{X}_{\gamma i}) \\ &= \sum_{\gamma=1}^{\Gamma} R_{\gamma i}\big(\boldsymbol{E}_{p\gamma\_m}(\mathbf{1}_{K\gamma\times 1} - \boldsymbol{X}_{\gamma i}) + e_o + e_{rx}\boldsymbol{L}_{\gamma n}\boldsymbol{X}_{\gamma i}\big) \end{aligned} \tag{5.6}$$

In cluster based WSNs, the master node typically is in charge of all of its n slave nodes. It has to iterate n times to finish the tasks of all the slave nodes. Thus, the total energy cost of the master node for handling n slave nodes is formulated as:

$$\begin{aligned} E_m(\boldsymbol{X}_{11,\cdots,\Gamma 1}, \cdots, \boldsymbol{X}_{1n,\cdots,\Gamma n}) &= \sum_{i=1}^{n}\sum_{\gamma=1}^{\Gamma} R_{\gamma i}E_{m\gamma\_i}(\boldsymbol{X}_{\gamma i}) \\ &= \sum_{i=1}^{n}\sum_{\gamma=1}^{\Gamma} R_{\gamma i}\big(\boldsymbol{E}_{p\gamma\_m}(\mathbf{1}_{K\gamma\times 1} - \boldsymbol{X}_{\gamma i}) + e_o + e_{rx}\boldsymbol{L}_{\gamma n}\boldsymbol{X}_{\gamma i}\big) \end{aligned} \tag{5.7}$$

Consequently, the network lifetime NL, which is defined as the time until the first node dies, can be formulated as:

$$NL = min\Big\{\frac{Bat_m}{E_m(\boldsymbol{X}_{11,\cdots,\Gamma 1}, \cdots, \boldsymbol{X}_{1n,\cdots,\Gamma n})}, \frac{Bat_1}{E_1(\boldsymbol{X}_{11,\cdots,\Gamma 1})}, \cdots, \frac{Bat_n}{E_n(\boldsymbol{X}_{1n,\cdots,\Gamma n})}\Big\} \tag{5.8}$$

Then, maximizing the network lifetime by using static task allocation for condition triggered application can be modeled by a BILP problem as shown in Eq. (5.9).

$$\arg\min_{X_{\gamma i}} \ max\Big\{\frac{E_m(\boldsymbol{X}_{11,\cdots,\Gamma 1},\cdots,\boldsymbol{X}_{1n,\cdots,\Gamma n})}{Bat_m}, \frac{E_1(\boldsymbol{X}_{11,\cdots,\Gamma 1})}{Bat_1}, \cdots, \frac{E_n(\boldsymbol{X}_{1n,\cdots,\Gamma n})}{Bat_n}\Big\} \tag{5.9}$$

subject to:

$$1 \leq \boldsymbol{1}_{1\times K\gamma}\boldsymbol{X}_{\gamma i} \leq K\gamma - 1$$
$$\boldsymbol{B}_\gamma^T \boldsymbol{X}_{\gamma i} \geqq \boldsymbol{0}$$

Note that the partition cut $\boldsymbol{X}_{\gamma i}$ is a time invariant binary vector. The optimal task allocation solutions can be obtained by solving the above static condition triggered task allocation (SCTTA) algorithm, i.e., Eq. (5.9).

Although the static task allocation solutions are able to efficiently extend the network lifetime, the dynamic task allocation using multiple partition cuts performs better as analyzed in Chapter 4. To this end, a dynamic condition triggered task allocation (DCTTA) is further proposed to achieve longer network lifetime. Assuming that slave node i applies different partition cuts when it executes DAG γ and the network collapses after T scheduling rounds, the mean energy cost of slave node i and the master node for executing DAG γ in each round are:

$$\overline{E}_{\gamma i} = \frac{1}{TR_{\gamma i}} \sum_{j=1}^{TR_{\gamma i}} E_{\gamma i}(\boldsymbol{X}_{\gamma i}^j) \tag{5.10}$$

$$\overline{E}_m = \frac{1}{TR_{\gamma i}} \sum_{j=1}^{TR_{\gamma i}} \sum_{i=1}^{n} E_{m\gamma\_i}(\boldsymbol{X}_{\gamma i}^j) \tag{5.11}$$

where $\boldsymbol{X}_{\gamma i}^j$ is the partition cut when slave node i and the master node execute stationary DAG γ in the j-th scheduling round. According to Eqs. (5.3) and (5.4), Eqs. (5.10) and (5.11) can be reformed by:

$$\begin{aligned}\overline{E}_i(\chi_{\gamma i}) &= \boldsymbol{E}_{p\gamma\_i}\frac{1}{TR_{\gamma i}}\sum_{j=1}^{TR_{\gamma i}} \boldsymbol{X}_{\gamma i}^j + e_o + e_{tx}(d_i)\boldsymbol{L}_{\gamma n}\frac{1}{TR_{\gamma i}}\sum_{j=1}^{TR_{\gamma i}} \boldsymbol{X}_{\gamma i}^j \\ &= \boldsymbol{E}_{p\gamma\_i}\chi_{\gamma i} + e_o + e_{tx}(d_i)\boldsymbol{L}_{\gamma n}\chi_{\gamma i}\end{aligned} \tag{5.12}$$

$$\begin{aligned}\overline{E}_m(\chi_{\gamma i}) &= \boldsymbol{E}_{p\gamma\_m}(\boldsymbol{1}_{K1\times 1} - \frac{1}{TR_{\gamma i}}\sum_{j=1}^{TR_{\gamma i}} \boldsymbol{X}_{\gamma i}^j) + e_o + e_{rx}\boldsymbol{L}_{\gamma n}\frac{1}{TR_{\gamma i}}\sum_{j=1}^{TR_{\gamma i}} \boldsymbol{X}_{\gamma i}^j \\ &= \boldsymbol{E}_{p\gamma\_m}(\boldsymbol{1}_{K2\times 1} - \chi_{\gamma i}) + e_o + e_{rx}\boldsymbol{L}_{\gamma n}\chi_{\gamma i}\end{aligned} \tag{5.13}$$

where $\chi_{\gamma i} = \frac{1}{TR_{\gamma i}} \sum_{j=1}^{TR_{\gamma i}} \boldsymbol{X}_{\gamma i}^{j}$ which can be treated as probability vector variable, since each element in $\boldsymbol{X}_{\gamma i}^{j}$ is within the range of [0, 1]. Based on the execution probability of each stationary DAG graph, the average energy cost of slave node i for executing every stationary DAG graph can be written as:

$$\begin{aligned}\overline{E}_i(\chi_{1i,\cdots,\Gamma i}) &= \sum_{\gamma=1}^{\Gamma} R_{\gamma i}\overline{E}_i(\chi_{\gamma i}) \\ &= \sum_{\gamma=1}^{\Gamma} R_{\gamma i}\big(\boldsymbol{E}_{p\gamma\_i}\chi_{\gamma i} + e_o + e_{tx}(d_i)\boldsymbol{L}_{\gamma n}\chi_{\gamma i}\big)\end{aligned} \tag{5.14}$$

Since the master node has to iterate n times for its n slave nodes, according to Eq. (5.13), the corresponding energy cost of the master node is formulated as:

$$\begin{aligned}\overline{E}_m(\chi_{11,\cdots,\Gamma 1}, \cdots, \chi_{1n,\cdots,\Gamma n}) &= \sum_{i=1}^{n}\sum_{\gamma=1}^{\Gamma} R_{\gamma i}\overline{E}_m(\chi_{\gamma i}) \\ &= \sum_{i=1}^{n}\sum_{\gamma=1}^{\Gamma} R_{\gamma i}\big(\boldsymbol{E}_{p\gamma\_m}(\boldsymbol{1}_{K\gamma\times 1} - \boldsymbol{X}_{\gamma i}) + e_o + e_{rx}\boldsymbol{L}_{\gamma n}\chi_{\gamma i}\big)\end{aligned} \tag{5.15}$$

By using $\chi_{\gamma i}$, the constraints for the partition cuts, Eqs. (5.1) and (5.2), can be rewritten as:

$$1 \le \boldsymbol{1}_{1\times K\gamma}\chi_{\gamma i} \le K\gamma - 1 \tag{5.16}$$

$$\boldsymbol{B}_{\gamma}^{T}\chi_{\gamma i} \geqq \boldsymbol{0} \tag{5.17}$$

Based on Eqs. (5.14) to (5.17) and the definition of the network lifetime NL, the dynamic condition triggered task allocation (DCTTA) for maximizing NL can be formulated by a LP problem as shown in Eq. (5.18).

$$\underset{\chi_{1i},\chi_{2i}}{\arg\min}\ max\left\{\frac{\overline{E}_m(\chi_{11,\cdots,\Gamma 1}, \cdots, \chi_{1n,\cdots,\Gamma n})}{Bat_m}, \frac{\overline{E}_1(\chi_{11,\cdots,\Gamma 1})}{Bat_1}, \cdots, \frac{\overline{E}_n(\chi_{1n,\cdots,\Gamma n})}{Bat_n}\right\} \tag{5.18}$$

subject to:

$$1 \le \boldsymbol{1}_{1\times K\gamma}\chi_{\gamma i} \le K\gamma - 1$$

$$\boldsymbol{B}_{\gamma}^{T}\chi_{\gamma i} \geqq \boldsymbol{0}$$

The task allocation solutions, $\chi_{\gamma i}$, can be obtained by solving the above DCTTA algorithm, i.e., Eq. (5.18). Each of them consists of at most two partition cuts with the corresponding weights, which has already been proved by Section 4.5.2. The detailed partition cuts and their weights can be calculated by Algorithm 1 as presented in Section 4.4.2.

In addition to the scenario of condition triggered tasks, the task allocation algorithms for joint local and global tasks will be presented in the next section.

5.2.2 Task Allocation Algorithms for Joint Local and Global Tasks: SJTA and DJTA

In this part, the modeling of the task allocation problem for the joint local and global tasks is firstly presented. Then, the static joint task allocation (SJTA) algorithm and the dynamic joint task allocation (DJTA) algorithm are described.

Modeling Joint Task Allocation Problem

For WSN applications which consist of joint local and global tasks as those presented in [24, 97], the slave and the master nodes are required to collaborate together to complete the global tasks periodically in addition to complete the local tasks with its master node. The model of the joint task allocation problem is made up of two parts: modeling the local task allocation and the global task allocation.

The local task allocation is the same as the modeling presented in Section 4.3.1 of Chapter 4. Each slave node completes the local application with the collaboration of its master node. Distributing the tasks of the local application is equivalent to divide the modeled local DAG graph into two subgraphs, $\boldsymbol{G}_s = (\boldsymbol{V}_s, \boldsymbol{E}_s)$ and $\boldsymbol{G}_m = (\boldsymbol{V}_m, \boldsymbol{E}_m)$, with a partition cut $\boldsymbol{X}$ as illustrated in Fig. 4.1. The transmitted data from the slave node to its master node is the summation of the weights of edges which cross the partition cut. Moreover, the constraints, Eqs. (4.1) and (4.2) have to be satisfied to make sure that the workload is completed by the cooperation of the slave and master and to guarantee that the data is transmitted from the slave to the master.

For the periodically operated global application, the task allocation problem consists in dividing the global DAG graph into multiple subgraphs and mapping them to the slave and master nodes. As Fig. 5.2 shows, the subgraph assigned to slave node i is constructed by two partition cuts: the *beginning cut* $\boldsymbol{X}_{bi}$ and the *ending cut* $\boldsymbol{X}_{ei}$. Considering that the slave nodes only communicate with its master node. The global application typically starts from the master node and also ends at the master node. Each slave node firstly receives the data from the master node then executes the assigned tasks of the global application, and finally transmits the data to the master node. The received data and the transmitted data of slave node i are the summations of the weights of edges which cross the *beginning cut* $\boldsymbol{X}_{bi}$ and the *ending cut* $\boldsymbol{X}_{ei}$, respectively. Taking Fig. 5.2 for example, the slave node i receives $l(e_6) + l(e_7)$ bits of data and transmits

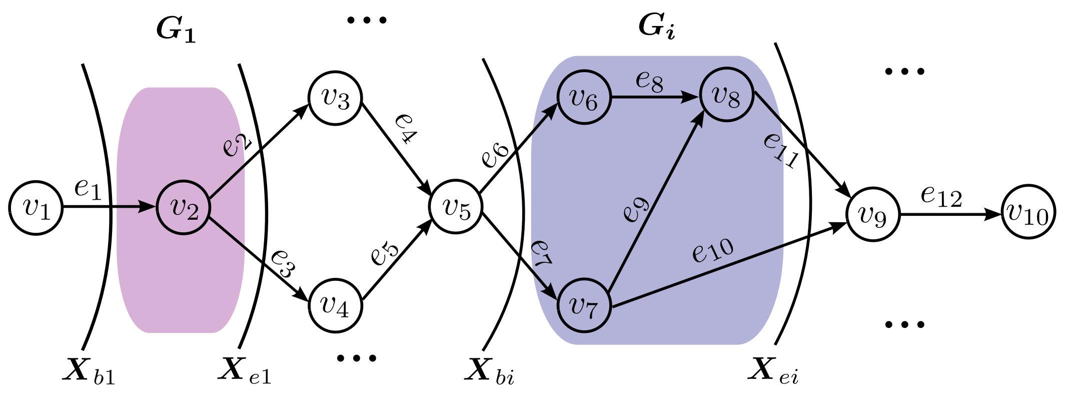


Figure 5.2: Schematic diagram of task allocation for global application.

$l(e_{10}) + l(e_{11})$ bits of data from and to the master node, respectively.

During each global task allocation round, each task of the global application has to be assigned to a single slave node. In other words, there should be no intersection among the subgraphs that are assigned to the slave nodes, which could be formulated as follows:

$$G_i \cap G_j = \varnothing \; (i, j = 1, \cdots, n; i \neq j) \tag{5.19}$$

where n is the number of the slave nodes. Moreover, the *beginning cut* X_{bi} is always in front of the *ending cut* X_{ei} unless they are the same which means no task assigned to slave i.

In order to maximize the network lifetime, it is necessary to consider the task allocation problems of both the local and global applications. The key issue is to find the optimal partition solutions of both the local and global DAG graphs to balance the workload distribution of the slave and master nodes.

Proposed Algorithms: SJTA and DJTA

This section presents the SJTA and DJTA algorithms by formulating the task allocation problems of both the local and global applications as a BILP and a LP problems, respectively. The proposed algorithms are run by the gateway which is assumed to be very powerful and without the energy limits.

Similar to the formulations in Section 4.3.2 of Chapter 4, $X = [x(v_1), \cdots, x(v_k), \cdots, x(v_K)]^T$ is used to represent the partition cut of the local application. K is the number of vertexes of the local DAG graph, and $x(v_k)$ is a boolean parameter which indicates whether the vertex v_k belongs to the slave node or the master node. $x(v_k)$ equals 1 if v_k belongs to the slave node and 0 otherwise. The constraints for the local partition cut are the same as formulated by Eqs. (4.4) and (4.5). Moreover, based on Eq. (4.6), the net generated data of each task in the local application is represented by $\boldsymbol{L}_l = [l_n(v_1), \cdots, l_n(v_k), \cdots, l_n(v_K)]$. Thus, the amount of the transmitted data from slave node i to the master node can be expressed as $\boldsymbol{L}_l\boldsymbol{X}_i$. The energy cost of slave node i for the local application consists of cost of executing the assigned local tasks and transmitting the data to the master, which can be formulated as:

$$E_{li}(\boldsymbol{X_i}) = \boldsymbol{E}_{l\_pi}\boldsymbol{X}_i + e_o + e_{tx}(d_i)\boldsymbol{L}_l\boldsymbol{X}_i \tag{5.20}$$

where $\boldsymbol{E}_{l\_pi} = [E_{pi}(v_1), \cdots, E_{pi}(v_k), \cdots, E_{pi}(v_K)]$ represents the processing energy cost of each local task when they are executed by slave node i. As the master node is in charge of its n slave nodes, it has to iterate n times to complete the local applications for the slave nodes in each local scheduling round. The corresponding energy cost of the master node is:

$$E_{lm}(\boldsymbol{X}_1, \cdots, \boldsymbol{X}_n) = \sum_{i=1}^{n} \Big(\boldsymbol{E}_{l\_pm}(\boldsymbol{1}_{K\times1} - \boldsymbol{X_i}) + e_o + e_{rx}\boldsymbol{L}_l\boldsymbol{X_i}\Big) \tag{5.21}$$

where $\boldsymbol{E}_{l\_pm} = [E_{pm}(v_1), \cdots, E_{pm}(v_k), \cdots, E_{pm}(v_K)]$ represents the processing cost of the master node for executing each local task, and $\boldsymbol{1}_{K\times1}$ is an $K \times 1$ all one vector.

Next, the energy cost of the slave and master node for executing the global tasks are formulated as follows. The beginning and the ending partition cuts are represented by two $H \times 1$ binary vectors, $\boldsymbol{X}_b = [x_b(v_1), \cdots, x_b(v_h), \cdots, x_b(v_H)]$ and $\boldsymbol{X}_e = [x_e(v_1), \cdots, x_e(v_h), \cdots, x_e(v_H)]$. H is the number of the vertexes in the global DAG. $x_b(v_h)$ and $x_e(v_h)$ are the boolean parameters defined by:

$$x_b(v_h) = \begin{cases} 1, & \text{if } v_h \text{ is in front of partition cut } \boldsymbol{X}_b \\ 0, & \text{otherwise} \end{cases} \tag{5.22}$$

and

$$x_e(v_h) = \begin{cases} 1, & \text{if } v_h \text{ is in front of partition cut } \boldsymbol{X}_e \\ 0, & \text{otherwise} \end{cases} \tag{5.23}$$

Considering Fig. 5.2 as one example, the beginning partition cut and the ending partition cut for slave node i are:

$$\boldsymbol{X}_{bi} = [1, 1, 1, 1, 1, 0, 0, 0, 0, 0]^T$$
$$\boldsymbol{X}_{ei} = [1, 1, 1, 1, 1, 1, 1, 1, 0, 0]^T$$

By using these two binary vector variables, the constraint of the no intersection among the subgraphs of the global application Eq. (5.19) is rewritten as:

$$\sum_{i=1}^{n}(\boldsymbol{X}_{ei} - \boldsymbol{X}_{bi}) \leqq \mathbf{1}_{H\times1} \tag{5.24}$$

where $\mathbf{1}_{H\times1}$ is an $H \times 1$ all one vector. Correspondingly, the constraint of the relation between the beginning and ending cuts can be expressed by:

$$\boldsymbol{X}_{bi} \leqq \boldsymbol{X}_{ei} \tag{5.25}$$

Let $\boldsymbol{L}_g = [l_n(v_1), \cdots, l_n(v_h), \cdots, l_n(v_H)]$ stand for the net generated data of each global task, the received and transmitted data of slave node i for executing global tasks can be formulated as $\boldsymbol{L}_g\boldsymbol{X}_{bi}$ and $\boldsymbol{L}_g\boldsymbol{X}_{ei}$, respectively.

The energy related activities of slave node i for the global application include receiving data from the master node, executing the assigned tasks and transmitting data to the master node. The corresponding energy cost of slave node i for the global tasks are formulated as:

$$E_{gi}(\boldsymbol{X}_{bi}, \boldsymbol{X}_{ei}) = e_o + e_{rx}\boldsymbol{L}_g\boldsymbol{X}_{bi} + \boldsymbol{E}_{g\_pi}(\boldsymbol{X}_{ei} - \boldsymbol{X}_{bi}) + e_o + e_{tx}(d_i)\boldsymbol{L}_g\boldsymbol{X}_{ei} \tag{5.26}$$

where $\boldsymbol{E}_{g\_pi} = [E_{pi}(v_1), \cdots, E_{pi}(v_h), \cdots, E_{pi}(v_H)]$ represents the processing energy cost of slave node i when it executes each vertex of the global DAG.

The energy cost of the master node for the global application consists of the energy consumed for executing the tasks that are not assigned to the slave nodes and communicating with the slave nodes. The corresponding formulation is shown as follows:

$$\begin{aligned} E_{gm}(\boldsymbol{X}_{b1\cdots bn}, \boldsymbol{X}_{e1\cdots en}) = {} & \boldsymbol{E}_{g\_pm}\big(\mathbf{1}_{H\times1} - \sum_{i=1}^{n}(\boldsymbol{X}_{ei} - \boldsymbol{X}_{bi})\big) \\ & + \sum_{i=1}^{n}(e_o + e_{tx}(d_i)\boldsymbol{L}_g\boldsymbol{X}_{bi}) + \sum_{i=1}^{n}(e_o + e_{rx}\boldsymbol{L}_g\boldsymbol{X}_{ei}) \end{aligned} \tag{5.27}$$

where $\boldsymbol{E}_{g\_pm} = [E_{pm}(v_1), \cdots, E_{pm}(v_h), \cdots, E_{pm}(v_H)]$ stands for the energy cost of the master node for executing each tasks of the global application.

The network lifetime, NL, of a cluster based WSN with 1 master and n slave nodes, according to the NL definition, can be expressed by:

$$NL = min\Big\{\frac{Bat_m(T+1)}{TE_{lm}(\boldsymbol{X}_{1,\cdots,n}) + E_{gm}(\boldsymbol{X}_{b1\cdots n}, \boldsymbol{X}_{e1\cdots n})}, \frac{Bat_i(T+1)}{TE_{li}(\boldsymbol{X}_i) + E_{gi}(\boldsymbol{X}_{bi}, \boldsymbol{X}_{ei})}\Big\},\ i = 1\cdots n \tag{5.28}$$

where T is the global application execution period, which means the global DAG has to be executed every T local task scheduling rounds. Consequently, maximizing NL by the joint task allocation can be formulated by a BILP problem as follows:

$$\arg \min_{\boldsymbol{X}_i, \boldsymbol{X}_{bi}, \boldsymbol{X}_{ei}} max\left\{\frac{TE_{lm}(\boldsymbol{X}_{1,\cdots,n}) + E_{gm}(\boldsymbol{X}_{b1\cdots n}, \boldsymbol{X}_{e1\cdots n})}{Bat_m}, \frac{TE_{li}(\boldsymbol{X}_i) + E_{gi}(\boldsymbol{X}_{bi}, \boldsymbol{X}_{ei})}{Bat_i}\right\} \tag{5.29}$$

subject to:

$$1 \leq \mathbf{1}_{1\times K}\boldsymbol{X}_i \leq K-1$$
$$\boldsymbol{B}^T\boldsymbol{X}_i \geqq \mathbf{0}$$
$$\sum_{i=1}^{n}(\boldsymbol{X}_{ei} - \boldsymbol{X}_{bi}) \leqq \mathbf{1}_{H\times 1}$$
$$\boldsymbol{X}_{bi} \leqq \boldsymbol{X}_{ei}$$

where Bat_m and Bat_i are the battery energy of the master node and slave node i, respectively. Note that, the partition cuts, $\boldsymbol{X}_i$, $\boldsymbol{X}_{bi}$ and $\boldsymbol{X}_{ei}$, are time invariant binary vectors, which are used to distribute the tasks statically for the slave and master nodes. Under this static policy, the optimal partition solutions can be obtained by solving the above static joint task allocation (SJTA) algorithm.

Since the dynamic task allocation achieves a more balanced workload distribution among the slave and master nodes as analyzed in Section 4.4.1, the SJTA algorithm is further extended to a dynamic joint task allocation (DJTA) algorithm by using multiple partition cuts for both the local and global tasks.

Considering that the slave and master nodes apply different local and global partition cuts in each local DAG execution round and global DAG execution round. Let $\boldsymbol{X}_i^j$, $\boldsymbol{X}_{bi}^j$ and $\boldsymbol{X}_{ei}^j$ denote the partition cuts that slave node i exploits in the j-th local task scheduling round and global task scheduling round, respectively. Assuming that the network elapses after $J(T+1)$ joint rounds, maximizing the network lifetime is equivalent to find appropriate $\boldsymbol{X}_i^j$, $\boldsymbol{X}_{bi}^j$ and $\boldsymbol{X}_{ei}^j$, $i = 1, \cdots, n$ and $j = 1, \cdots, T$, to maximize $J(T+1)$. This problem can be formulated as:

$$\arg \max_{\boldsymbol{X}_i^j, \boldsymbol{X}_{bi}^j, \boldsymbol{X}_{ei}^j} \quad J(T+1) \tag{5.30}$$

subject to:

$$\sum_{j=1}^{JT} E_{li}(\boldsymbol{X}_i^j) + \sum_{j=1}^{J} E_{gi}(\boldsymbol{X}_{bi}^j, \boldsymbol{X}_{ei}^j) \leq Bat_i, \; i = 1, \cdots, n$$

$$\sum_{j=1}^{JT} E_{lm}(\boldsymbol{X}_1^j, \cdots, \boldsymbol{X}_n^j) + \sum_{j=1}^{J} E_{gm}(\boldsymbol{X}_{b1\cdots n}^j, \boldsymbol{X}_{e1\cdots n}^j) \leq Bat_m$$

$$1 \leq \mathbf{1}_{1\times K}\boldsymbol{X}_i \leq K-1$$

$$\boldsymbol{B}^T\boldsymbol{X}_i \geqq \mathbf{0}$$

$$\sum_{i=1}^{n}(\boldsymbol{X}_{ei} - \boldsymbol{X}_{bi}) \leqq \mathbf{1}_{H\times 1}$$

$$\boldsymbol{X}_{bi} \leqq \boldsymbol{X}_{ei}$$

Still, it is very complicated for the dynamic task allocation to calculate the $nJT + nJ$ variables, since the network lifetime typically lasts for hundreds of thousands of the joint local and global rounds.

The proposed DJTA reduces the complexity and formulates the dynamic local and global task allocation problem as a LP problem based on the average energy cost of the nodes. According to Eqs. (5.20) and (5.21), the average energy cost of slave node i and the master node in each local DAG execution round, $\overline{E}_{li}$ and $\overline{E}_{lm}$, are:

$$\begin{aligned}\overline{E}_{li} &= \frac{1}{JT}\sum_{j=1}^{JT} E_{li}(\boldsymbol{X}_i^j) \\ &= e_o + (\boldsymbol{E}_{l\_pi} + e_{tx}(d_i)\boldsymbol{L}_l)\frac{1}{JT}\sum_{j=1}^{JT}\boldsymbol{X}_i^j \end{aligned} \tag{5.31}$$

$$\begin{aligned}\overline{E}_{lm} &= \frac{1}{JT}\sum_{j=1}^{JT} E_{lm}(\boldsymbol{X}_1^j, \cdots, \boldsymbol{X}_n^j) \\ &= \sum_{i=1}^{n}\left(\boldsymbol{E}_{l\_pm}\mathbf{1}_{K\times 1} + e_o + (e_{rx}\boldsymbol{L}_l - \boldsymbol{E}_{l\_pm})\frac{1}{JT}\sum_{j=1}^{T}\boldsymbol{X}_i^j\right)\end{aligned} \tag{5.32}$$

Let $\chi_i = \frac{1}{JT}\sum_{j=1}^{JT}\boldsymbol{X}_i^j = [\chi_i(v_1), \cdots \chi_i(v_k), \cdots \chi_i(v_K)]^T$, in which each element $\chi_i(v_k) = \frac{1}{JT}\sum_{j=1}^{JT} x_i^j(v_k)$ indicates how often the local task v_k is executed by the slave node i. Based on χ_i, Eqs. (5.31) and (5.32) are reformed as follows:

$$\overline{E}_{li}(\chi_i) = e_o + \left(\boldsymbol{E}_{l\_pi} + e_{tx}(d_i)\boldsymbol{L}_l\right)\chi_i \tag{5.33}$$

$$\overline{E}_{lm}(\chi_1, \cdots, \chi_n) = \sum_{i=1}^{n}\left(\boldsymbol{E}_{l\_pm}\mathbf{1}_{K\times 1} + e_o + (e_{rx}\boldsymbol{L}_l - \boldsymbol{E}_{l\_pm})\chi_i\right) \tag{5.34}$$

Correspondingly, according to Eqs. (5.26) and (5.27), the average energy cost of slave node i

and the master node in each global DAG execution round can be formulated as:

$$\overline{E}_{gi}(\chi_{bi}, \chi_{ei}) = \frac{1}{J}\sum_{j=1}^{J} E_{gi}(\boldsymbol{X}_{bi}^{j}, \boldsymbol{X}_{ei}^{j}) \tag{5.35}$$

$$= 2e_o + (e_r x\boldsymbol{L}_g - \boldsymbol{E}_{g\_pi})\chi_{bi} + (\boldsymbol{E}_{g\_pi} + e_{tx}(d_i)\boldsymbol{L}_g)\chi_{ei}$$

$$\overline{E}_{gm}(\chi_{b1\cdots bn}, \chi_{e1\cdots en}) = \frac{1}{J}\sum_{j=1}^{J} E_{gm}(\boldsymbol{X}_{b1\cdots bn}^{j}, \boldsymbol{X}_{e1\cdots en}^{j}) \tag{5.36}$$

$$= \boldsymbol{E}_{g\_pm}\big(\boldsymbol{1}_{H\times 1} - \sum_{i=1}^{n}(\chi_{ei} - \chi_{bi})\big) + \sum_{i=1}^{n}(2e_o + e_{tx}(d_i)\boldsymbol{L}_g\chi_{bi} + e_{rx}\boldsymbol{L}_g\chi_{ei})$$

where $\chi_{bi} = [\chi_{bi}(v_1), \cdots, \chi_{bi}(v_h), \cdots, \chi_{bi}(v_H)]$ and $\chi_{ei} = [\chi_{ei}(v_1), \cdots, \chi_{ei}(v_h), \cdots, \chi_{ei}(v_H)]$. The elements in χ_{bi} and χ_{ei} satisfy $\chi_{bi}(v_h) = \frac{1}{J}\sum_{j=1}^{J} x_{bi}^{j}(v_h)$ and $\chi_{ei}(v_h) = \frac{1}{J}\sum_{j=1}^{J} x_{ei}^{j}(v_h)$, which are used together to represent how often the global task is assigned to the slave node i.

By using the three probability vector variables χ_i, χ_{bi} and χ_{ei}, the constraints of the local and global tasks can be reformed by:

$$1 \leq \boldsymbol{1}_{1\times K}\chi_i \leq K - 1 \tag{5.37}$$

$$\boldsymbol{B}^T\chi_i \geqq \boldsymbol{0} \tag{5.38}$$

$$\sum_{i=1}^{n}(\chi_{ei} - \chi_{bi}) \leqq \boldsymbol{1}_{H\times 1} \tag{5.39}$$

$$\chi_{bi} \leqq \chi_{ei} \tag{5.40}$$

Based on Eqs. (5.33) to (5.40), DJTA for maximizing the network lifetime of a cluster based WSN which consists of 1 master node and n slave nodes can be formulated by a LP problem as follows:

$$\arg\min_{\chi_i, \chi_{bi}, \chi_{ei}} max\left\{\frac{TE_{lm}(\chi_{1,\cdots,n}) + E_{gm}(\chi_{b1\cdots n}, \chi_{e1\cdots n})}{Bat_m}, \frac{TE_{li}(\chi_i) + E_{gi}(\chi_{bi}, \chi_{ei})}{Bat_i}\right\} \tag{5.41}$$

subject to:

$$1 \leq \boldsymbol{1}_{1\times K}\chi_i \leq K - 1$$

$$\boldsymbol{B}^T\chi_i \geqq \boldsymbol{0}$$

$$\sum_{i=1}^{n}(\chi_{ei} - \chi_{bi}) \leqq \boldsymbol{1}_{H\times 1}$$

$$\chi_{bi} \leqq \chi_{ei}$$

By comparing with SJTA which provides the static partition solutions, the partition solutions obtained by DJTA are made up of probabilities and indicate multiple partition cuts with the different weights. After obtaining χ, χ_b and χ_e by solving the above LP problem, the detailed partition cuts and the corresponding weights can be calculated by Algorithm 1 as presented in Section 4.4.2 of Chapter 4.

5.3 Extensions of Task Allocation for Multi-hop Mesh WSNs: DTA-mhop

In addition to the cluster based WSNs, the multi-hop mesh WSNs have also been widely studied and applied to real applications, e.g, [98]. The task allocation algorithms for cluster based WSNs presented in Chapter 4 are not sufficient for those cases. Thus, this section extends the previous algorithms for the multi-hop mesh network architecture.

As presented in Section 2.2.1. In a multi-hop mesh network, all the WSN nodes (sensor nodes) are considered equal with respect to their roles and functionalities. For generality, the sensor nodes are considered to have different characteristics, such as battery energy, processing power or even the transmitting and receiving power, while they are defined to have the same transmission ranges. Each sensor node can directly communicate with any of the nodes which are within the maximum transmission range, i.e., its neighbor nodes. In addition to the abilities of sensing, processing and transmitting, the sensor nodes also can operate as the relay nodes. The observation of each sensor node is propagated by multiple wireless hops from the first relay node to the next one until the observation reaches the destination (sink node). Note that each sensor node is an individual source and has to execute its own application. All of the neighbor nodes of the sender are able to act as the relay nodes, which can share the tasks of the senders as well. The selection of the transmission path depends on the used routing protocol.

5.3.1 Modeling the Task Allocation for Multi-hop Mesh WSNs

In the above introduced multi-hop mesh WSNs, each sensor node not only has to execute the tasks of its own application, but also it needs to consider whether to share part of the tasks of its neighbor nodes or just forward the received data. Since each sensor node is considered as an individual source, its own application has to be completed by the sensor node itself, the relay nodes and the sink node together. Consequently, the task allocation for the application of each sensor node is to distribute the whole workload to the sensor node, the relay nodes and the sink node, respectively.

Considering a simple scenario of the multi-hop mesh network as shown in Fig. 5.3, the data produced by sensor node 1 needs to pass sensor node 2 and 3 and then reaches the sink node; sensor node 3 connects with the sink node by one wireless hop and performs as the relay node of sensor node 2 as well. Let $\boldsymbol{X} = [x(v_1), \cdots, x(v_k), \cdots, x(v_K)]^T$ represent the partition solution of the application for each sensor node. K is the number of the vertexes in the DAG graph and $x(v_k)$ is a boolean parameter which equals 1 when vertex v_k is assigned to the sensor node and 0 otherwise, respectively.

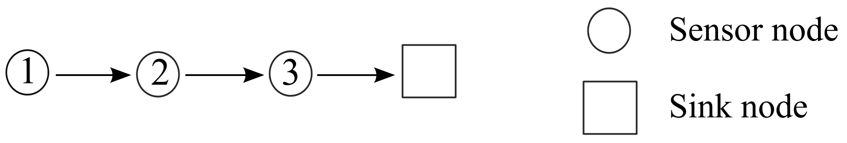


Figure 5.3: A simple scenario of the multi-hop mesh WSN (the solid arrows represent the transmission directions).

Fig. 5.4 illustrates the partition solutions of the application of each sensor node. For the application of sensor node 1, the corresponding partition solutions for sensor node 1, 2, 3 and the sink node are $\boldsymbol{X}_{11}$, $\boldsymbol{X}_{12}$, $\boldsymbol{X}_{13}$ and $\mathbf{1}_{K\times1} - \boldsymbol{X}_{11} - \boldsymbol{X}_{12} - \boldsymbol{X}_{13}$, respectively. Iteratively, the partition solutions of the application of sensor node 2 are made up of $\boldsymbol{X}_{21}$, $\boldsymbol{X}_{22}$, and $\mathbf{1}_{K\times1} - \boldsymbol{X}_{21} - \boldsymbol{X}_{22}$; $\boldsymbol{X}_{31}$ and $\mathbf{1}_{K\times1} - \boldsymbol{X}_{31}$ construct the partition solutions of the application of sensor node 3.

According to the above model of task allocation for the multi-hop mesh WSNs, the corresponding task allocation algorithm is presented in the next section.

5.3.2 DTA-mhop Algorithm

For the clear description of the proposed algorithm, this section firstly presents a simple case and then gives the algorithm for the general networks.

Simple Multi-hop Network

The task allocation algorithm for the multi-hop network is introduced by considering the simple network scenario as depicted in Fig. 5.3. By studying this simple network, the main concept of the algorithm is illustrated. The more general networks will be considered in the next sub-section.

For simplicity, it considers that the energy cost activities of the sensor nodes are mainly related to the processing and communicating cost. Let $\boldsymbol{E}_{pi} = [E_{pi}(v_1), \cdots, E_{pi}(v_k), \cdots, E_{pi}(v_K)]$ denote the processing energy cost of each vertex when they are executed by sensor node i and $\boldsymbol{L}_n = [l_n(v_1), \cdots, l_n(v_k), \cdots, l_n(v_K)]$ represent the net generated data of each vertex in the DAG

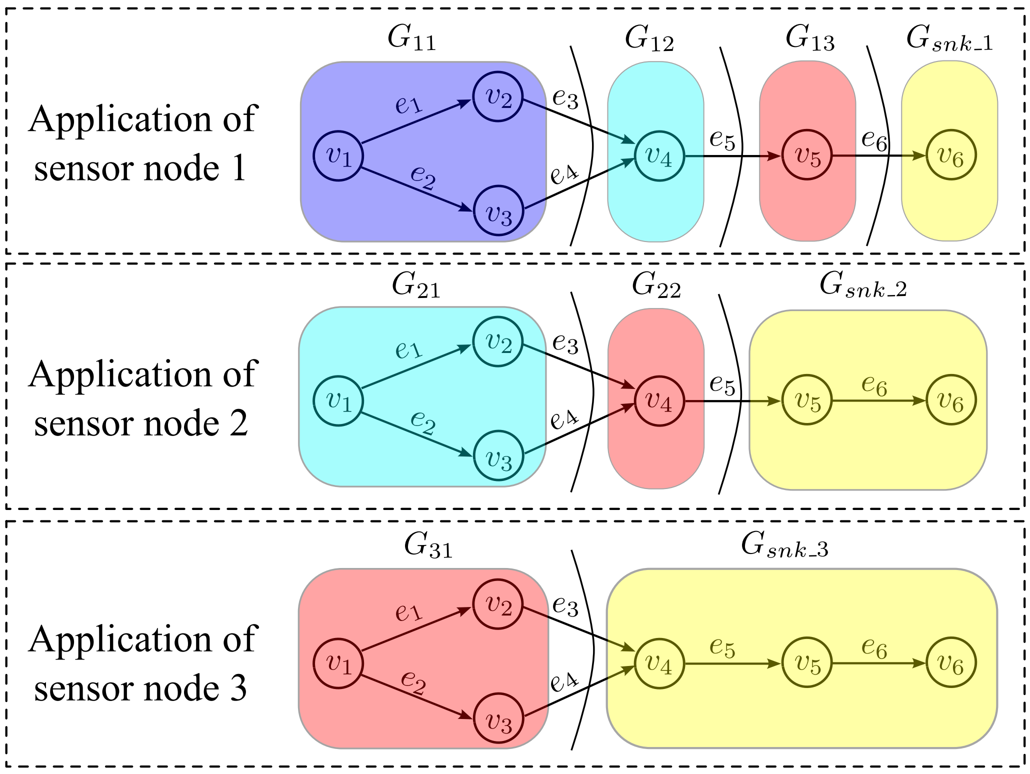

Application of sensor node 1
Application of sensor node 2
Application of sensor node 3
G_{11}
G_{12}
G_{13}
G_{snk_1}
G_{21}
G_{22}
G_{snk_2}
G_{31}
G_{snk_3}
v_1 v_2 v_3 v_4 v_5 v_6
e_1 e_2 e_3 e_4 e_5 e_6

Figure 5.4: Schematic diagram of task allocation for the multi-hop mesh network

graph, respectively. According the modeling presented in Section 5.3.1, sensor node 1 spends energy on processing the assigned workload of its own application and transmitting data to sensor node 2. The energy cost of sensor node 1 can be formulated as:

$$E_1 = \boldsymbol{E}_{p1}\boldsymbol{X}_{11} + e_o + e_{tx}(d_1)\boldsymbol{L}_n\boldsymbol{X}_{11} \tag{5.42}$$

Like sensor node 1, sensor node 2 has to process the workload of its own application and transmit to sensor node 3. In addition, sensor node 2 also needs to receive the data from sensor node 1 and decide to share parts of its workload or not, and then transmit to sensor node 3. Thus, the energy cost of sensor node 2 can be expressed as:

$$\begin{aligned} E_2 =& \boldsymbol{E}_{p2}\boldsymbol{X}_{21} + e_o + e_{tx}(d_2)\boldsymbol{L}_n\boldsymbol{X}_{21} \\ &+ e_o + e_{rx}\boldsymbol{L}_n\boldsymbol{X}_{11} + \boldsymbol{E}_{p2}\boldsymbol{X}_{12} + e_o + e_{tx}(d_2)\boldsymbol{L}_n(\boldsymbol{X}_{11} + \boldsymbol{X}_{12}) \end{aligned} \tag{5.43}$$

Note that $\boldsymbol{X}_{12}$ can be an all zero vector, which means sensor node 2 does not share any of the tasks for sensor node 1 and just forwards the received data. Sensor node 3 has to pass the data directly or indirectly for sensor node 2 and sensor node 1, in addition to execute the tasks of its own application. Similar to Eq. (5.43), the energy cost of sensor node 3 is:

$$\begin{aligned}E_3 =&\boldsymbol{E}_{p3}\boldsymbol{X}_{31} + e_o + e_{tx}(d_3)\boldsymbol{L}_n\boldsymbol{X}_{31}\\ &+ e_o + e_{rx}\boldsymbol{L}_n(\boldsymbol{X}_{11} + \boldsymbol{X}_{12}) + \boldsymbol{E}_{p3}\boldsymbol{X}_{13} + e_o + e_{tx}(d_3)\boldsymbol{L}_n(\boldsymbol{X}_{11} + \boldsymbol{X}_{12} + \boldsymbol{X}_{13})\\ &+ e_o + e_{rx}\boldsymbol{L}_n\boldsymbol{X}_{21} + \boldsymbol{E}_{p3}\boldsymbol{X}_{22} + e_o + e_{tx}(d_3)\boldsymbol{L}_n(\boldsymbol{X}_{21} + \boldsymbol{X}_{22})\end{aligned} \tag{5.44}$$

The sink node is in charge of receiving the data and completing the rest of the application of each sensor node. Its energy cost is then formulated as:

$$\begin{aligned}E_{snk} =&e_o + e_{rx}\boldsymbol{L}_n(\boldsymbol{X}_{11} + \boldsymbol{X}_{12} + \boldsymbol{X}_{13}) + \boldsymbol{E}_{psnk}(\boldsymbol{1}_{K\times 1} - \boldsymbol{X}_{11} - \boldsymbol{X}_{12} - \boldsymbol{X}_{13})\\ &+ e_o + e_{rx}\boldsymbol{L}_n(\boldsymbol{X}_{21} + \boldsymbol{X}_{22}) + \boldsymbol{E}_{psnk}(\boldsymbol{1}_{K\times 1} - \boldsymbol{X}_{21} - \boldsymbol{X}_{22})\\ &+ e_o + e_{rx}\boldsymbol{L}_n\boldsymbol{X}_{31} + \boldsymbol{E}_{psnk}(\boldsymbol{1}_{K\times 1} - \boldsymbol{X}_{31})\end{aligned} \tag{5.45}$$

Since the dynamic task allocation using multiple partition cuts is better than the static task allocation as presented in Chapter 4, this section mainly focuses on the dynamic task allocation. Firstly, the mean energy cost of the sensor nodes and the sink node in each scheduling round are exploited. Similar to Section 4.4.1, the static binary partition solutions, $\boldsymbol{X}_{11}$, $\boldsymbol{X}_{12}$, $\boldsymbol{X}_{13}$, $\boldsymbol{X}_{21}$, $\boldsymbol{X}_{22}$ and $\boldsymbol{X}_{31}$ are replaced by the probability partition solutions, $\chi_{11}, \chi_{12}, \chi_{13}, \chi_{21}, \chi_{22}$ and χ_{31}, respectively. Moreover, as there are multiple partitions for the application of each sensor node, the summation of all of the partition solutions for each sensor node is introduced to reduce the number of the variables and thereby reducing the complexity. Let $\mathbb{X}_1 = \chi_{11}$, $\mathbb{X}_2 = \chi_{21} + \chi_{12}$ and $\mathbb{X}_3 = \chi_{31} + \chi_{22} + \chi_{13}$. According to Eqs. (5.42) to (5.45), the average energy cost of the sensor nodes and the sink node in each scheduling round can be expressed as:

$$\overline{E}_1(\mathbb{X}_1) =\boldsymbol{E}_{p1}\mathbb{X}_1 + e_o + e_{tx}(d_1)\boldsymbol{L}_n\mathbb{X}_1 \tag{5.46}$$

$$\overline{E}_2(\mathbb{X}_1, \mathbb{X}_2) =e_o + e_{rx}\boldsymbol{L}_n\mathbb{X}_1 + e_o + e_{tx}(d_2)\boldsymbol{L}_n\mathbb{X}_1 + \boldsymbol{E}_{p2}\mathbb{X}_2 + e_o + e_{tx}(d_2)\boldsymbol{L}_n\mathbb{X}_2 \tag{5.47}$$

$$\begin{aligned}\overline{E}_3(\mathbb{X}_1, \mathbb{X}_2, \mathbb{X}_3) =&e_o + e_{rx}\boldsymbol{L}_n\mathbb{X}_1 + e_o + e_{tx}(d_3)\boldsymbol{L}_n\mathbb{X}_1 + e_o + e_{rx}\boldsymbol{L}_n\mathbb{X}_2\\ &+ e_o + e_{tx}(d_3)\boldsymbol{L}_n\mathbb{X}_2 + \boldsymbol{E}_{p3}\mathbb{X}_3 + e_o + e_{tx}(d_3)\boldsymbol{L}_n\mathbb{X}_3\end{aligned} \tag{5.48}$$

$$\overline{E}_{snk}(\mathbb{X}_1, \mathbb{X}_2, \mathbb{X}_3) =3\boldsymbol{E}_{psnk}\boldsymbol{1}_{K\times 1} - \boldsymbol{E}_{psnk}(\mathbb{X}_1 + \mathbb{X}_2 + \mathbb{X}_3) + 3e_o + e_{rx}\boldsymbol{L}_n(\mathbb{X}_1 + \mathbb{X}_2 + \mathbb{X}_3) \tag{5.49}$$

Based on the mean energy cost functions in each scheduling round, the lifetime of this simple multiple hop network, NL, can be represented by:

$$NL = min\Big\{\frac{Bat_{snk}}{\overline{E}_{snk}(\mathbb{X}_1, \mathbb{X}_2, \mathbb{X}_3)}, \frac{Bat_1}{\overline{E}_1(\mathbb{X}_1)}, \frac{Bat_2}{\overline{E}_2(\mathbb{X}_1, \mathbb{X}_2)}, \frac{Bat_3}{\overline{E}_3(\mathbb{X}_1, \mathbb{X}_2, \mathbb{X}_3)}\Big\} \tag{5.50}$$

Consequently, the problem of maximizing NL is modeled as the following LP problem.

$$\arg\min_{\mathbb{X}_1,\mathbb{X}_2,\mathbb{X}_3} max\left\{\frac{\overline{E}_{snk}(\mathbb{X}_1,\mathbb{X}_2,\mathbb{X}_3)}{Bat_{snk}}, \frac{\overline{E}_1(\mathbb{X}_1)}{Bat_1}, \frac{\overline{E}_2(\mathbb{X}_1,\mathbb{X}_2)}{Bat_2}, \frac{\overline{E}_3(\mathbb{X}_1,\mathbb{X}_2,\mathbb{X}_3)}{Bat_3}\right\} \tag{5.51}$$

Subject to:

$$\boldsymbol{B}^T(\mathbf{1}_{K\times1}-\mathbb{X}_1) \leqq \mathbf{0},\ \boldsymbol{B}^T(\mathbf{2}_{K\times1}-\mathbb{X}_1-\mathbb{X}_2) \leqq \mathbf{0},\ \boldsymbol{B}^T(\mathbf{3}_{K\times1}-\mathbb{X}_1-\mathbb{X}_2-\mathbb{X}_3) \leqq \mathbf{0}; \tag{5.52}$$

$$\mathbb{X}_1(1)=1, \mathbb{X}_1(end)=0,\ \mathbb{X}_2(1)=1, \mathbb{X}_2(end)=0,\ \mathbb{X}_3(1)=1, \mathbb{X}_3(end)=0; \tag{5.53}$$

$$\mathbf{0} \leqq \mathbb{X}_1 \leqq \mathbf{1}_{K\times1}, \mathbf{0} \leqq \mathbb{X}_2 \leqq \mathbf{2}_{K\times1}, \mathbf{0} \leqq \mathbb{X}_3 \leqq \mathbf{3}_{K\times1}. \tag{5.54}$$

where $\mathbf{2}_{K\times1}$ and $\mathbf{3}_{K\times1}$ are $K \times 1$ vectors and each elements of them equal 2 and 3, respectively; Eq. (5.52) is the constraint to guarantee the transmitted data is from the source node to the sink node; Eq. (5.53) is the constraint to ensure the first task (sensing task) is done by the sensor node and the data is finally received by the sink node.

The summations of the partition solutions for the sensor nodes, $\mathbb{X}_1$, $\mathbb{X}_2$ and $\mathbb{X}_3$, can be obtained by solving the above LP problem in the gateway which typically has powerful computing ability and no energy constraint. The corresponding partition solutions for each sensor node are calculated as follows: the partition solution for sensor node 1, $\boldsymbol{X}_{11}$, equals $\mathbb{X}_1$; then, the partition solutions of sensor node 2 are calculated based on $\boldsymbol{X}_{11}$, $\mathbb{X}_2$ and the constraints of the partition solutions; finally, the partition solutions of sensor node 3 can be obtained based on the partition solutions of sensor node 1 and 2.

After illustrating the task allocation algorithm for a simple multi-hop WSN, the general multi-hop networs are considered in the next part.

General Multi-hop Network

In this section, a general multi-hop WSN, as depicted in Fig. 2.2 of Section 2.2.1, is considered to illustrate the task allocation algorithm. For such a network, the minimum hop routing algorithm, which is one of the most popular routing algorithms in existing literature (e.g., [36, 37, 38, 39]), is employed for each sensor node to find the path to the sink node. According to this routing algorithm, the network can be represented by a static tree structure WSN as shown in Fig. 5.5.

The relationships of the sensor nodes based on this tree structure are expressed by an inter-node adjacency matrix $\boldsymbol{A}$, whose rows and columns stand for the sensor nodes in order, respectively. Each element of $\boldsymbol{A}$ is defined by:

$$\boldsymbol{A}(i,j) = \begin{cases} 1, & \text{if sensor node } i \text{ is on the path from sensor node } j \text{ to sink node} \\ 0, & \text{otherwise} \end{cases}$$

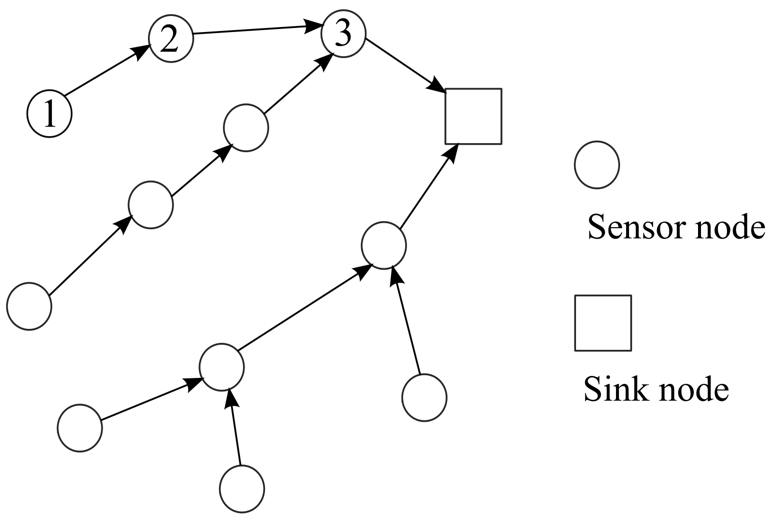


Figure 5.5: The tree structure model for a general multi-hop WSN as depicted in Fig. 2.2 of Section 2.2.1 by using the minimum hop routing algorithm.

Note that sensor node i is not considered on the path from itself to the sink node. In other words, $\boldsymbol{A}_{ij} = 0$ when $i = j$. Thus, the inter-node adjacency matrix $\boldsymbol{A}$ for Fig. 5.5 is:

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ 1 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

After modeling the tree structure, the energy cost of each sensor node is formulated. According to Eqs. (5.46) to (5.48), the average energy cost of sensor node i in each scheduling round can be formulated as:

$$\begin{aligned}\overline{E}_i(\mathbb{X}_j, \mathbb{X}_i) = &\boldsymbol{E}_{pi}\mathbb{X}_i + e_o + e_{tx}(d_i)\boldsymbol{L}_n\mathbb{X}_i \\ &+ \sum_{j \in \boldsymbol{C}_i} \big(e_o + e_{rx}\boldsymbol{L}_n\mathbb{X}_j + e_o + e_{tx}(d_i)\boldsymbol{L}_n\mathbb{X}_j\big)\end{aligned} \tag{5.55}$$

where $\boldsymbol{C}_i$ is the set of sensor nodes whose paths to the sink node pass sensor node i, which can be represented by the rows of the inter-node adjacency matrix $\boldsymbol{A}$. Correspondingly, based on Eq. (5.49), the average energy cost of the sink node in each scheduling round is expressed by:

$$\overline{E}_{snk}(\mathbb{X}_1, \cdots, \mathbb{X}_n) = n\boldsymbol{E}_{psnk}\mathbf{1}_{K\times 1} + \sum_{i=1}^{n} \big(e_o + e_{rx}\boldsymbol{L}_n\mathbb{X}_i - \boldsymbol{E}_{psnk}\mathbb{X}_i\big) \tag{5.56}$$

where n is the number of the sensor nodes in the network.

Therefore, the dynamic task allocation algorithms for maximizing the lifetime of a general multi-hop mesh WSN, DTA-mhop, can be formulated as the following LP problem:

$$\arg\min_{\mathbb{X}_i} \quad max\left\{\frac{\overline{E}_{snk}(\mathbb{X}_1,\cdots,\mathbb{X}_n)}{Bat_{snk}}, \frac{\overline{E}_i}{Bat_i}\,\middle|\, i=1,\cdots,n\right\} \tag{5.57}$$

Subject to:

$$\boldsymbol{B}^T\big((N_c+1)\mathbf{1}_{K\times 1} - \mathbb{X}_i - \sum_{j\in\boldsymbol{C}_i}\mathbb{X}_j\big) \leqq \mathbf{0};$$
$$\mathbb{X}_i(1)=1, \mathbb{X}_i(end)=0;$$
$$\mathbf{0} \leqq \mathbb{X}_i \leqq (N_c+1)\mathbf{1}_{K\times 1}.$$

where N_c stands for the number of the sensor nodes in $\boldsymbol{C}_i$. According to the obtained summation of the partition solutions of each sensor node, the detailed partition solutions for the sensor nodes can be calculated iteratively.

5.4 Simulation Results

This section illustrates the performances of the proposed task allocation algorithms for different task scenarios and network structures based on simulation results of the artificially generated scenarios. The network lifetime increase with respect to the no-scheduling strategy and the algorithm execution time are investigated.

5.4.1 Simulation Setup

For the cluster based WSNs, the slave and the master nodes are considered to use the same CC2538 system-on-chip, and the RF module works at the $2.4GHz$ ISM band with the bandwidth of 250 $Kbps$. According to the datasheet of CC2538 system-on-chip, the detailed information of the energy parameters are listed in Table 5.1. As each cluster can work independently, the proposed algorithms for scenarios of the condition triggered tasks and the joint local and global tasks are carried out within the cluster. To make the simulation environments close to the realistic scenarios, the cluster is randomly generated in a two dimension area of 100×100 square meters with one master node and n slave nodes. The n slave nodes are randomly distributed in the area and the master node is located at the center. The battery energy of the slave nodes are within the range of $[1kJ, 5kJ]$ while the master node has a battery within the range of $[6kJ, 10kJ]$.

For the multi-hop mesh networks, the energy parameters for the sensor nodes are also obtained

Table 5.1: Values of the energy related parameters used in the artificially generated scenarios according to CC2538 system-on-chip datasheet.

Processing energy cost parameters	f_i, MCU processing speed of slave node i	32 MHz
	f_m, MCU processing speed of the master node	32 MHz
	P_i, processing power of slave node i	36.9 mW
	P_m, processing power of the master node	36.9 mW
Communication energy cost parameters	e_o, communication overhead energy cost	3.69 uJ
	t_{rx}, the time of RF for receiving 1 bit of data packet	4 μs
	t_{tx}, the time of RF for transmitting 1 bit of data packet	4 μs
	P_{T0}, energy cost of electronic circuits of the RF for receiving or transmitting 1 bit of data packet	59.8 mW
	P_{rin}, receiver sensitivity	-85 dBm
	η, drain efficiency	0.05
	F, RF frequency	2.4 GHz
	α, the path loss exponent	2

from CC2538 system-on-chip datasheet as illustrated in Table 5.1. While the sink node is considered to incorporate a Texas Instruments TMS320C5509A as a dedicated DSP processor in addition to the CC2538. The sink node uses the transceiver subsystem in CC2538 for executing communication tasks. According to the datasheet of TMS320C5509A [99], the processing speed of the sink node is 200 MHz and the processing power is 192 mW. The network is randomly generated in a two dimension area of 200×200 square meters with one sink node and n sensor nodes. The maximum transmission range of each sensor node is 40 meters. Each sensor node transmits data to the sink node based on the minimum hop routing protocol. The battery energy of the sensor nodes and the sink node are distributed within the ranges of $[1kJ, 5kJ]$ and $[10kJ, 20kJ]$, respectively.

Each reported simulation result corresponds to the average values and the standard deviations of 500 test instances.

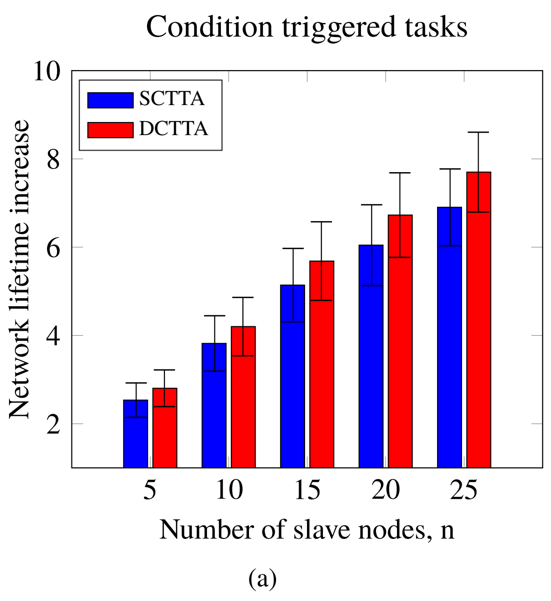
5.4.2 Evaluation of SCTTA and DCTTA Algorithms

Based on the energy parameters illustrated in Section 5.4.1, this section evaluates the performance of the condition triggered task allocation algorithms according to 2 metrics: network lifetime increase with respect to the no-scheduling strategy, in which each slave node only executes its first task and then directly transmits data to the master node; and the algorithm runtime (measured by running them in Matlab 2017a). The DAG graphs with conditional branches for the condition triggered applications are randomly generated according to the number of the conditions. The number of the vertexes of each divided stationary DAG graph is within the range of [10, 15]. The computation workload of each vertex is distributed within the range of [100, 1000] kilo clock cycles (KCCs). The amount of the communicated data on the edges are distributed within the range of [100, 500] bits. The network lifetime increase with respect to the no-scheduling strategy and the algorithm runtime are investigated by changing: a) The number of slave nodes, n; b) The variation among the tasks; c) The number of divided stationary DAG graphs. Among the parameters, the variation levels are defined as: 1) Low variation level, the workload of the tasks are randomly generated; 2) Middle variation level, randomly select one task and enlarge its workload 10 times based on the low level; 3) High variation level: randomly select one task and enlarge its workload 50 times based on the low level. The configuration parameters are summarized in Table 5.2, and only one parameter is changed in each experiment.

Table 5.2: Configuration parameters used in the simulations for estimation of condition triggered task allocation algorithms.

Parameters	Values	
	Default	Varied
Number of slave nodes, n	10	{5, 10, 15, 20, 25}
Variation level among the tasks	middle	{ low, middle, high}
Number of divided stationary DAGs	2	{ 1, 2, 3, 4, 5}

Fig. 5.6 depicts the network lifetime increase by using the proposed SCTTA and DCTTA algorithms with respect to the no-scheduling strategy and the corresponding algorithm runtime for the condition triggered tasks. It can be easily observed that the network lifetime increases by using both the SCTTA and DCTTA algorithms become more significant when the number of slave nodes n changes from 5 to 25. Taking the DCTTA algorithm for example, it extends the network lifetime in average from 2.81 to 7.70 times longer than the no-scheduling strategy. This is due to the fact that the task allocation algorithms can efficiently balance the workload among the slave and master nodes while the no-scheduling strategy makes the master node exponentially overloaded as n increases. Moreover, DCTTA algorithm prolongs the network lifetime longer than SCTTA. Its superiority becomes larger when n increases from 5 to 25. Since DCTTA provides multiple partition solutions for each sub-DAG graph, more balanced workload distribution can be achieved. This superiority accumulates as the total workload becomes heavier. In addition to the network lifetime increase, Fig. 5.6b shows the time requirements for executing the proposed algorithms. Both SCTTA and DCTTA need more execution runtime as n increases. This is because that the numbers of the variables in these two algorithms are related to n. For a large size WSN, it can be firstly grouped into small to medium sized clusters and



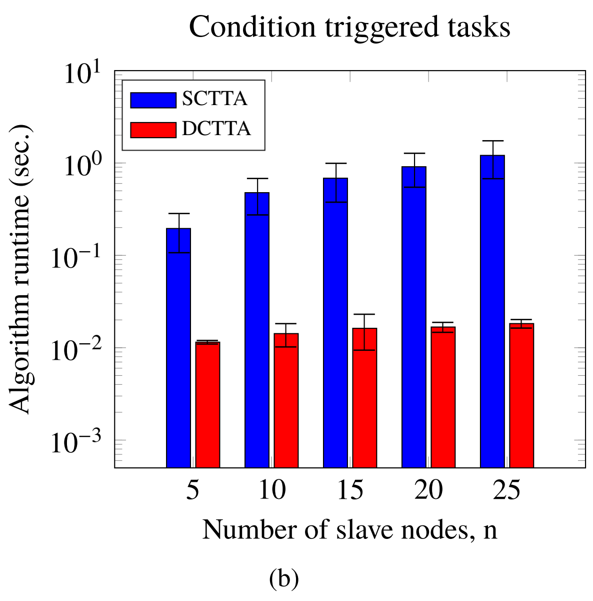
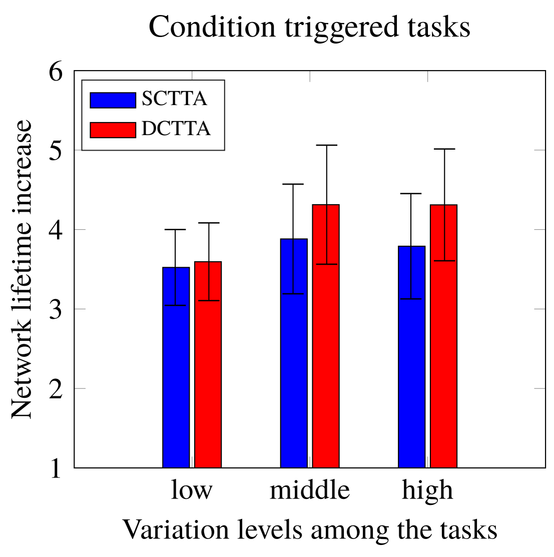


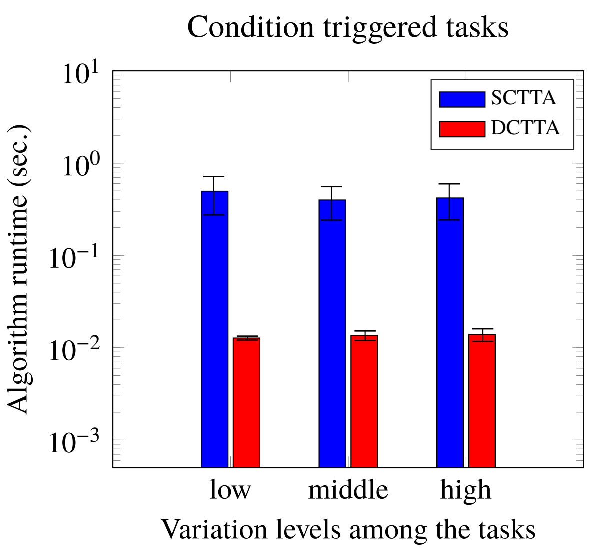
Figure 5.6: Effect of the number of slave nodes in the cluster on the (a) network lifetime increase and (b) algorithm runtime of SCTTA and DCTTA algorithms for the condition triggered tasks, respectively (The middle variation level among the tasks is selected and the number of conditions is 2, respectively).

then each cluster applies the algorithms independently.

Next, the effect of the variation among the tasks is estimated in this set of simulations. As depicted in Table 5.2, three variation levels are selected. Fig. 5.7a shows that the trends of the network lifetime improvement in average slightly go up by using SCTTA and DCTTA task allocation algorithms. Specifically, the DCTTA algorithm improves the network lifetime in average from 359.47% to 431.05% as the variation level changes from low to high. According to the definitions of the variation levels, the whole workload increases as the variation become higher. The task allocation algorithms efficiently balance the workload distribution among the slave and master nodes, while the no-scheduling strategy cannot. A more interesting phenomenon is that DCTTA extends the network lifetime longer than SCTTA, and the advantage of DCTTA increases as the variation level becomes higher. The reason is that the heavy tasks can be better shared among the slave and master nodes by using multiple partition cuts with the corresponding weights than using the static partition cuts. It is better to use DCTTA for tasks with high variation. Regarding the execution time of the proposed algorithms, it can be clearly seen in Fig. 5.7b that the algorithm runtime of both SCTTA and DCTTA remain stable. As the variation level changes from low to high, executing SCTTA and DCTTA need 0.45, 0.39, 0.41 and 0.013, 0.013, 0.014 seconds, respectively. This can be explained by the fact that the variation



(a)



(b)

Figure 5.7: Effect of the variation among the tasks on the (a) network lifetime increase and (b) algorithm runtime of SCTTA and DCTTA algorithms for the condition triggered tasks, respectively (There are $n = 10$ slave nodes in the cluster and the number of conditions is 2, respectively).

level does not increase the numbers of the variables, i.e., the complexities of the algorithms.

Further on, the performances of SCTTA and DCTTA algorithms are evaluated by changing the number of the conditions of the applications, Γ. Fig. 5.8a shows that the gain of the network lifetime increase for DCTTA remains stable as Γ changes from 1 to 5. The reason is that although the increasing Γ produces more sub-DAG graphs, each slave node only executes one sub-DAG, which does not affect the total workload too much. Different from DCTTA, the gain of SCTTA goes up as Γ increases and even SCTTA performs as well as DCTTA. Using one static task allocation solutions for the sub-DAG graphs cannot fairly balance the workload for each of them. While the whole workload of all of the subDAG graphs is more fairly balanced as the number of conditions increases. Moreover, the time requirements for executing both SCTTA and DCTTA increase as Γ becomes larger, since the complexities of the algorithms are related to both the number of slave nodes and the number of conditions. As shown in Fig. 5.8b, the algorithm runtime of DCTTA goes up in average from 0.012 to 0.029 seconds when Γ changes from 1 to 5. While the algorithm runtime of SCTTA increases from 0.127 to 7.067 seconds. Thus, for DAG graphs with large number of conditions, SCTTA requires much more execution time than DCTTA, which is a major limit for it to be used on-line.

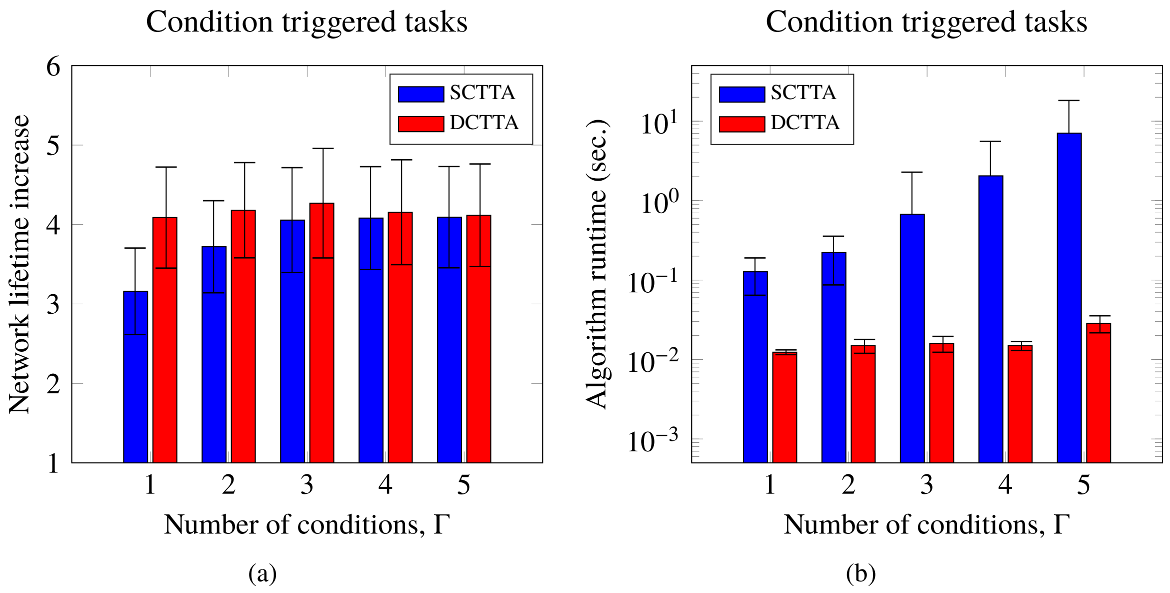


Figure 5.8: Effect of the number of conditions on the (a) network lifetime increase and (b) algorithm runtime of SCTTA and DCTTA algorithms for the condition triggered tasks, respectively (There are $n = 10$ slave nodes in the cluster and the middle level variation is selected).

5.4.3 Evaluation of SJTA and DJTA Algorithms

This section evaluates the performances of the proposed joint static and dynamic task allocation algorithms by comparing with the no-scheduling strategy and two other task allocation approaches:

- *No scheduling* strategy: The slave node only executes the first local tasks, the rest of the local tasks and the whole global tasks are done by the master node.
- Local Task Allocation (*Local TA*) [31]: It only provides the task solutions for the local application, the global tasks are fully completed by the master node itself.
- Global Task Allocation (*Global TA*): It only focuses on the global application task allocation, the first local task is executed by the slave node and the rest are done by the master node.

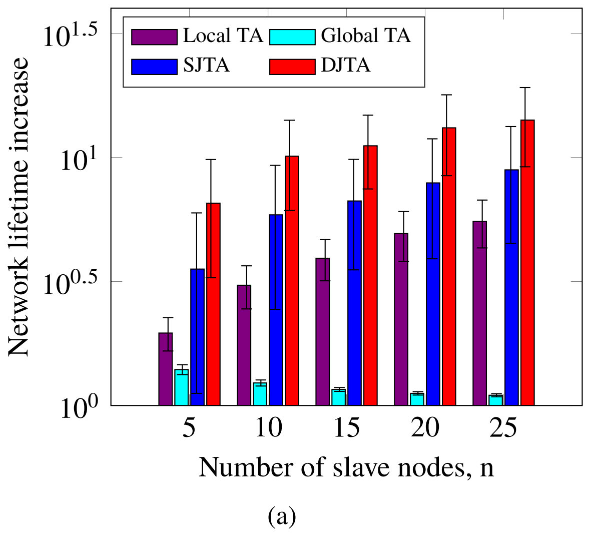
Note that both *Local TA* and *Global TA* approaches run within each cluster as well as our proposed algorithms. The computational workload of the vertexes in the local and global DAG graphs are randomly generated within the ranges of $[100, 500]$ KCCs and $[10000, 50000]$ KCCs, respectively. The amount of the communicated data on the edges of the local and global DAGs are distributed within the range of $[100, 500]$ bits. The increases of the network lifetime by using the task allocation algorithms with respect to the no-scheduling strategy and the execution time of running them in Matlab 2017a are investigated by changing: a) The number of slave nodes, n; b) The number of local tasks, K; c) The number of global tasks, H; d) The execution period of global tasks, T. The configuration parameters are summarized in Table 5.3, and only one parameter is changed in each experiment.

Table 5.3: Configuration parameters of the simulations for estimation of the joint local and global task allocation.

Parameters	Values	
	Default	Varied
Number of slave nodes, n	10	{5, 10, 15, 20, 25}
The number of local tasks, K	10	{5, 10, 15, 20}
The number of global tasks, H	20	{ 15, 20, 25, 3}
Global task execution period, T	200	{50, 100, 200, 500, 1000}

The first set of simulations is conducted to investigate the performances of the proposed task allocation algorithms by changing the number of slave nodes, n. Fig. 5.9a illustrates that the increases of the network lifetime by applying the, *Local TA*, SJTA and DJTA become more significant as n changes from 5 to 25. In contrast, the improvement of network lifetime by using *Global TA* decreases from 140% to 110% in average. The total workload of the local tasks of the slave nodes becomes larger as n increases, which makes the master node overburdened and die soon under no-scheduling strategy. While the energy consumption of the slave and master nodes are well balanced by *Local TA*, SJTA and DJTA. Moreover, the increase of the local task workload brings another consequence that only distributing the global tasks has less effect on extending the network lifetime. Due to the same reason, the superiority of DJTA and SJTA over *Local TA* gets smaller. Regarding the algorithm runtime, as shown in Fig. 5.9b, the four approaches require more time for running the algorithm as the number of slave nodes increases. This is a common limitation of the centralized algorithms. For a large sized WSN, the clustering approaches are needed to group the network into small clusters. The detailed procedure will be illustrated in Chapter 6.

The second set of simulations is to evaluate the algorithm performances when changing the number of tasks in the local application, K. It can be seen in Fig. 5.10a that the gains of improving the network lifetime by using *Local TA*, SJTA and DJTA are slightly increased. Since the local tasks become more complex as K increases, there are more task allocation possibilities for the approaches which take the local task into account. In contrast, the *Global TA* only focuses on the global tasks, which is not affected by changing the local tasks. Meanwhile, the increasing



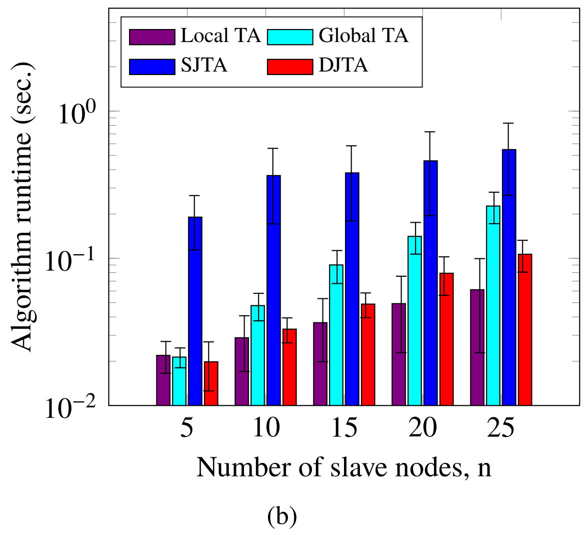
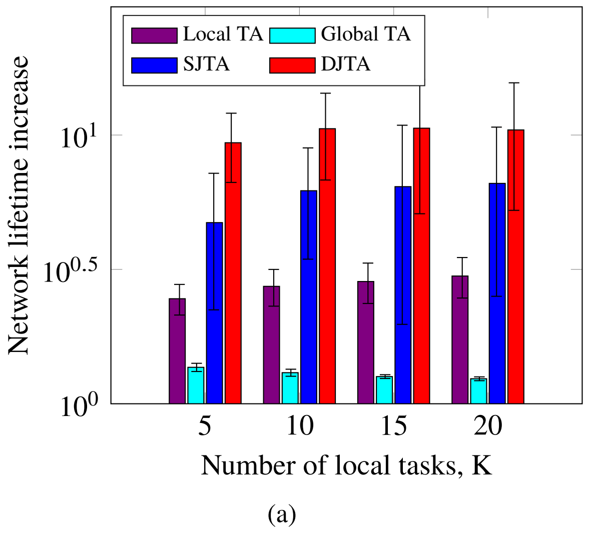


Figure 5.9: Effect of the number of the slave nodes in the cluster on (a) network lifetime increase and (b) algorithm runtime (there are 20 and 10 tasks in global and local applications, respectively, and T = 200 rounds).



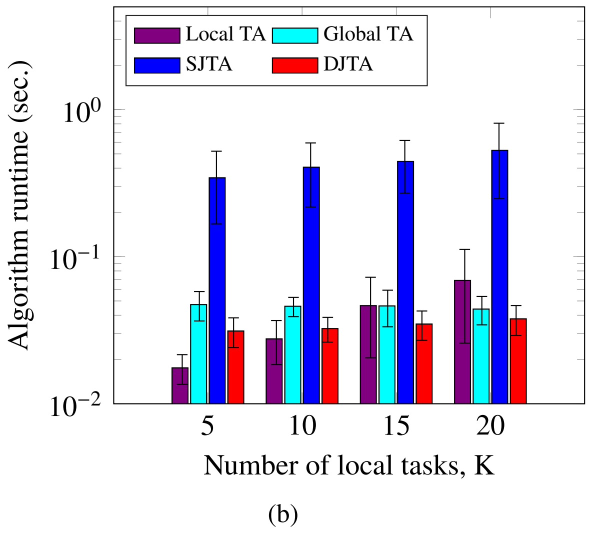


Figure 5.10: Effect of the number of tasks in local application on (a) network lifetime increase and (b) algorithm runtime (there are 10 slave nodes in the network and 20 tasks in global application, and T = 200 rounds).

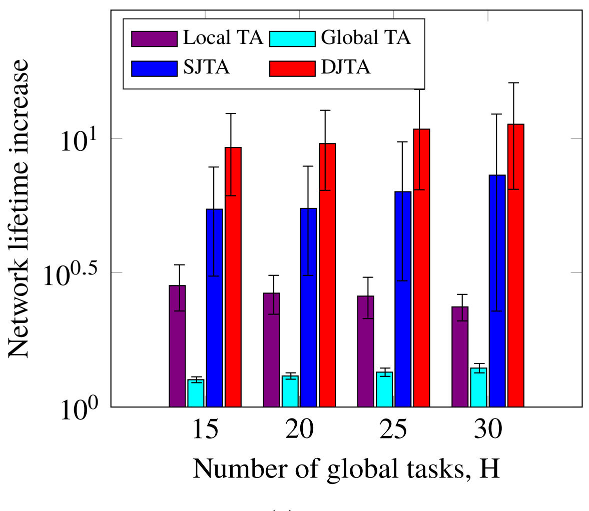
local tasks decreases the proportion of the global tasks in the whole local and global DAGs. The consequence is that the gain by using *Global TA* decreases from 136.63% to 123.89%. The corresponding algorithm runtime of *Global TA* is stable all the time while the *Local TA* consumes more time as K increases (see Fig. 5.10b). Although SJTA provides the static solution which is easier to implement in the slave and master nodes, DJTA achieves longer network lifetime and requires much less execution time. For example, when they are 20 local tasks, executing DJTA needs 0.037 seconds and achieves 1060.23% network lifetime improvement, while executing SJTA requires 0.528 seconds and extend the network lifetime 660.63% longer.

In addition to the local tasks, the impact of the number of global tasks, H, on the proposed algorithms is further estimated. The results are shown in Fig. 5.11. The gains of *Global TA*, SJTA and DJTA on extending the network lifetime over the no-scheduling strategy increase while the gain of *Local TA* is slightly decreasing. This is due to the fact that *Local TA* only focuses on the workload distribution of local tasks. Correspondingly, its algorithm runtime remains the same as H increases from 15 to 30, while the others spend more time on executing the algorithms (see Fig. 5.11b). Besides, DJTA performs better than SJTA on the network lifetime increase and execution time, which are consistent with the above-mentioned results. Note that, comparing with $T = 200$ rounds of local tasks, the workload of the global tasks is still relatively very small which leads to very small gains for *Global TA*. Thus, the next set of simulations adjusts the workload proportion of the global tasks by changing its execution period to evaluate the algorithm performance.

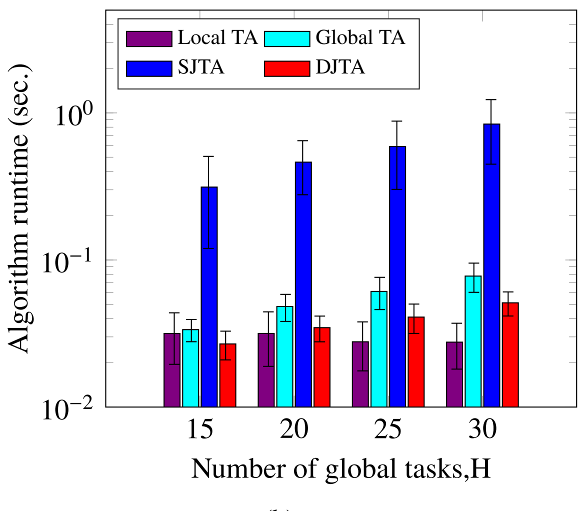
As the above depicted, changing the number of global tasks does not affect the algorithm too much when the execution period $T = 200$. The impact of T on the performance of the proposed algorithms is investigated in this part and the results are illustrated in Fig. 5.12. It is obvious that Global TA extends the network lifetime significantly when T is very small, i.e., the workload proportion of the global tasks is relatively large. It improves the network lifetime in average by 173% when $T = 50$. As the interval length of T increases, the performance of Global TA goes down. On the contrary, *Local TA* extends the network lifetime longer when T changes from 50 to 1000. Its performance is closer to SJTA and DJTA due to the decreasing global task workload. Since SJTA and DJTA take both the local and global tasks into account, they prolong the network lifetime dramatically all the time. Meanwhile, their algorithm execution times decrease, as the global task workload is reducing when T increases.

5.4.4 Evaluation of DTA-mhop Algorithm

This section evaluates the performance of the proposed task allocation algorithm for multi-hop mesh networks by comparing with the no-scheduling strategy and the task allocation algorithm

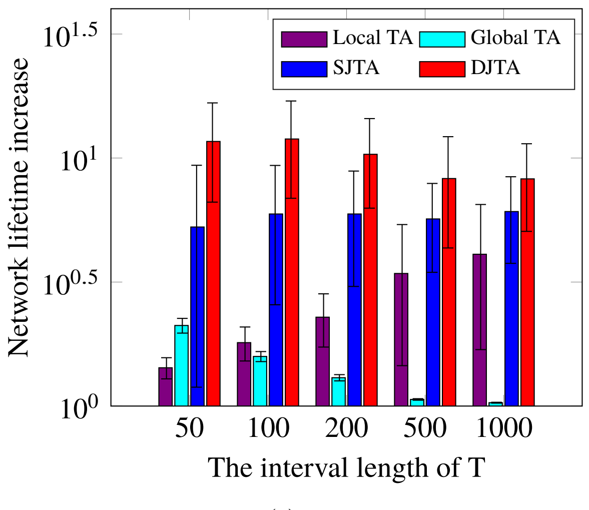


(a)

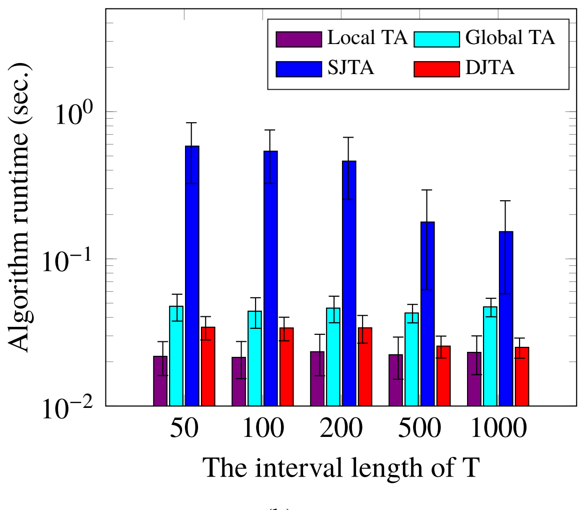


(b)

Figure 5.11: Effect of the number of the tasks in global application on (a) network lifetime increase and (b) algorithm runtime (there are 10 slave nodes in the network and 10 tasks in local application, and T = 200 rounds).



(a)



(b)

Figure 5.12: Effect of the execution period of the global application on (a) network lifetime increase and (b) algorithm runtime (there are 10 slave nodes in the network, 20 and 10 tasks in global and local applications, respectively).

for cluster based WSNs, CDTA:

- No scheduling strategy: each sensor node does not share any workload of its neighboring nodes and just executes the first task of itself.
- CDTA: firstly, CDTA algorithm is executed to obtain the task allocation solutions by considering that each sensor node connects with the sink node by one hop; then the sensor nodes apply the obtained task allocation solutions and just forward the data of their neighboring nodes.

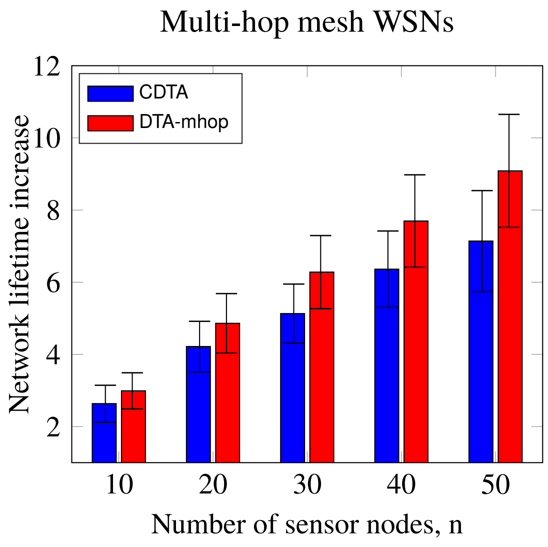
The increase of the network lifetime with respect to the no-scheduling strategy and the algorithm runtime are investigated by changing: a) The number of sensor nodes, n; b) The number of tasks, K. The configuration parameters are summarized in Table 5.4, and only one parameter is changed in each experiment.

Table 5.4: Configuration parameters in the simulations to evaluate the performance of the task allocation algorithm for multi-hop mesh networks.

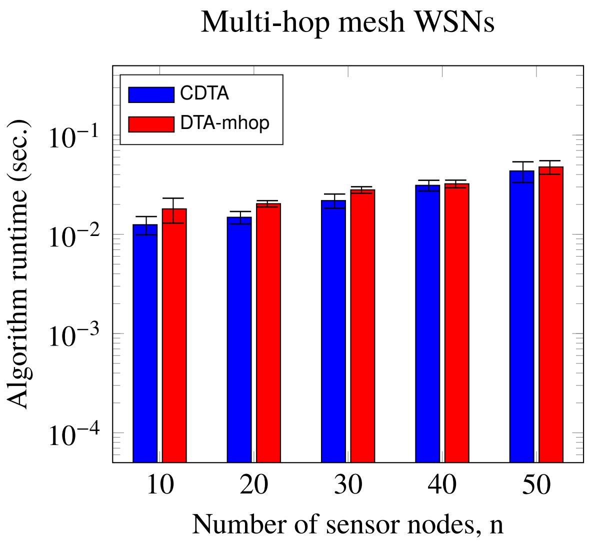
Parameters	Values	
	Default	Varied
Number of sensor nodes, n	20	{10, 20, 30, 40, 50}
The number of tasks, K	10	{5, 10, 15, 20}

Fig. 5.13 shows the effect of the number of the sensor nodes on the algorithm performance. It is obvious that the improvements of the network lifetime of both DTA-mhop and CDTA increase as the number of sensor nodes, n, changes from 10 to 50. As depicted in Fig. 5.13a, DTA-mhop achieves the gain of 299.08% network lifetime improvement when there are 10 sensor nodes while this gain increases to 908.61% when n equals 50. The proposed task allocation algorithms can efficiently balance the workload among the sensor nodes and the sink node, which makes the network last longer than using the no-scheduling strategy. Moreover, the DTA-mhop performs much better than CDTA, and the superiority of DTA-mhop becomes more significant as n increases. This is due to the fact that the CDTA considers the sensor nodes are connected with the sink node by one hop, which does not consider the energy cost of multi-hop transmission. As n increases, the workload of multi-hop transmission becomes heavier. Thus, it can be seen in Fig. 5.13a that DTA-mhop extends the network lifetime from 113.48% to 128.25% longer than CDTA when n equals 10 and 50, respectively. Further on, Fig. 5.13b shows that the time requirements for executing CDTA and DTA-mhop algorithms are very close and both of them increase as n changes from 10 to 50. Specifically, the algorithm runtime of DTA-mhop increases in average from 0.018 to 0.047 seconds and the runtime of CDTA goes up from 0.012 to 0.044

seconds. Since the DTA-mhop algorithm uses the summation of the task allocation solutions for each sensor node, its complexity is related to the number of sensor nodes as well as CDTA algorithm.



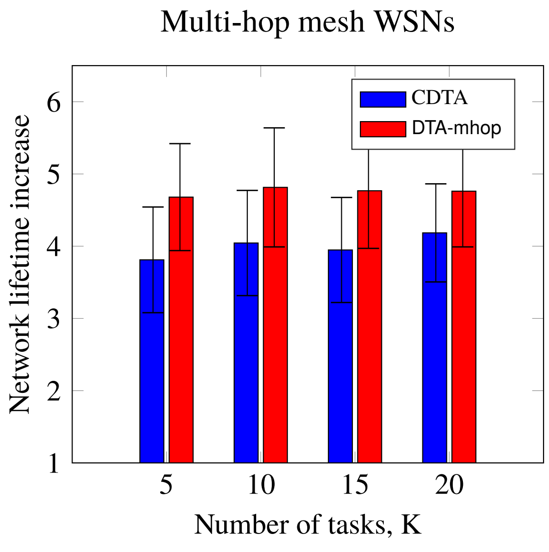
(a)



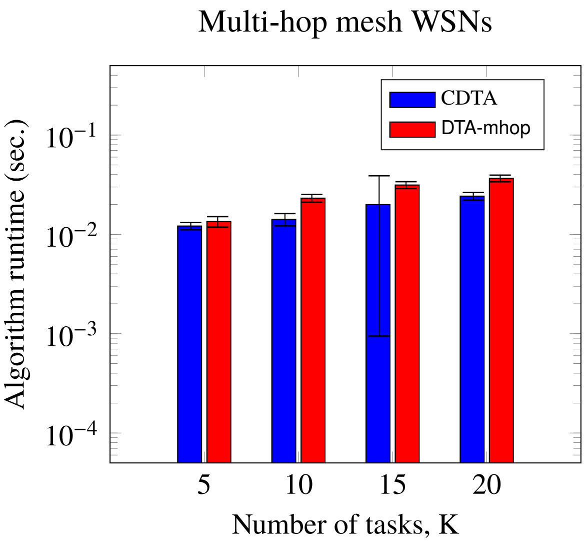
(b)

Figure 5.13: Effect of the number of sensor nodes on (a) network lifetime increase and (b) algorithm runtime of CDTA and the proposed DTA-mhop for multi-hop mesh WSNs (there are 10 tasks in each individual application).

In addition to estimate the effect of the number of sensor nodes, another set of simulations is conducted to investigate the algorithm performance by changing the number of the tasks. As demonstrated in Fig. 5.14a, the gains of the network lifetime improvement by using DTA-mhop and CDTA task allocation algorithms slightly become larger. Moreover, the superiority of DTA-mhop over CDTA decreases as the number of tasks increases, e.g., DTA-mhop extends the network lifetime in average from 122.79% to 113.8% longer than CDTA when the number of tasks changes from 5 to 20. This can be explained by the fact that the proportion of the workload of increased tasks becomes larger, which makes the effect the multi-hop transmission smaller. As the above analyzed that the complexities of DTA-mhop and CDTA are the same, their algorithm runtimes are still very close as shown in Fig. 5.14b. This phenomenon is consistent with the results in Fig. 5.13b.



(a)



(b)

Figure 5.14: Effect of the number of tasks in each application on (a) network lifetime increase and (b) algorithm runtime of CDTA and the proposed DTA-mhop for multi-hop mesh WSNs (there are 20 sensor nodes in the network).

5.5 Summary

This chapter has extended the previous task allocation algorithms proposed in Chapter 4 for different task scenarios and the multi-hop mesh network structure.

Firstly, the scenario of condition triggered tasks is considered. Given a specific condition triggered application, it can be modeled by a DAG graph with conditional branches. According to the conditional branches, i.e., the conditions of the tasks, the conditional DAG graph is classified into multiple stationary sub-DAG graphs without conditional branches. The task allocation problem for condition triggered tasks is then modeled as dividing each of the sub-DAG graphs into two parts simultaneously. The simulation results show that significant gains of the network lifetime increase with respect to the no-scheduling strategy have been achieved. The gains of both the SCTTA and DCTTA algorithms increase as the number of the slave nodes becomes larger. Moreover, the dynamic algorithm DCTTA extends the network lifetime longer than the static algorithm SCTTA as expected. For a cluster with 25 slave nodes, DCTTA extends the network lifetime 7.70 times longer w.r.t. no-scheduling strategy while SCTTA achieves 6.90 times longer network lifetime. Besides, for DAG graphs with large number of conditions, SCTTA requires much more execution time than DCTTA, which is a major limit for it to be used on-line.

Next, the task allocation algorithms for the joint local and global tasks are illustrated. Both the local and global applications are represented by normal DAG graphs. The task allocation problem for the local tasks is the same as the modeling in Chapter 4, while the global task allocation is modeled as dividing the global DAG graph into multiple subgraphs mapping to all of the slave and master nodes. Based on these modelings, a static and dynamic joint task allocation algorithms, SJTA and DJTA, are proposed based on the BILP and LP, receptively. Extensive simulations are investigated to estimate the performances of SJTA and DJTA on extending the network lifetime and the algorithm execution time. For a cluster with 10 slave nodes, SJTA and DJTA prolong the network lifetime 729.33% and 1128.09% longer w.r.t. no-scheduling strategy. Further on, the superiorities of the two algorithms become more significant for complex applications with large number of tasks. The characteristics of the proposed algorithms make them very suitable for computationally intensive WSN applications, especially in Internet of Things or in-network processing scenarios.

In addition to different task scenarios, the task allocation algorithm for the multi-hop mesh network structure is further studied. The sensor nodes have to not only execute their own assigned tasks but also to act as the relay nodes to forward the data of their neighboring nodes. Meanwhile, they are able to decide whether to just forward the data or share parts of the tasks of the neighboring nodes. The task allocation problem for multi-hop mesh networks is modeled by dividing the DAG graph of each sensor node into multiple subgraphs mapping to itself, the relay nodes and the sink node. In order to reduce the number of the variables, the summation of the task allocation solutions for each sensor node is used. Based on this scenario, DTA-mhop is proposed by formulating the problem of maximizing the network lifetime for multi-hop WSNs as a LP problem. The simulation results demonstrate that theDTA-mhop dramatically prolongs the network lifetime. It extends the network lifetime by 908.61% longer for a multi-hop network with 50 sensor nodes. Since the summation of the task allocation solutions for each sensor node is used, the complexity of the algorithm is only related to the number of the sensor nodes while has nothing to do with the complexity of the network structure. This characteristic makes the algorithm runtime of DTA-mhop very small: less than 0.05 seconds.

6 Experimental Results

6.1 Introduction **107**

6.2 Estimation Using Real WSN Applications **108**

6.2.1 Experiment Setup 108

6.2.2 Experimental Results 110

6.3 Hardware Implementation Using OpenMote-CC2538 Platform **116**

6.3.1 OpenMote-CC2538 116

6.3.2 Hardware Implementation Results 117

6.4 Summary **122**

6.1 Introduction

In previous chapters, the simulations of the task allocation algorithms proposed for cluster based WSNs are carried out within the cluster. This chapter aims to estimate the algorithm performance on general WSNs by using realistic applications and real hardware implementations.

The experimental evaluation is structured into two phases. Section 6.2 firstly investigates the algorithm performance using two real computation applications: *spectrum* computation and *maximum entropy power spectrum* (MEPS) computation. Targeting a general WSN, the network will be grouped into several small to medium sized clusters according to one of the most famous clustering protocol, LEACH [41]. Then, the task allocation algorithms are executed for each cluster individually, and the smallest lifetime among the clusters is chosen as the network lifetime. The performance of proposed algorithms on extending the network lifetime are compared with the no-scheduling strategy and the approach presented in [29, 100] that uses the same applications and network models. In the second phase, Section 6.3 presents the implementation of proposed algorithms on real WSN mote, the OpenMote-CC2538 platform [101], to measure the real gains in energy consumption and network lifetime improvement.

6.2 Estimation Using Real WSN Applications

This section evaluates the performance of the proposed CSTA, CDTA and DOOTA task allocation algorithms by using the real applications, spectrum and MEPS computations, on general WSN scenarios[1]. It firstly describes the detailed information of the two real applications in Section 6.2.1. Then, Section 6.2.2 presents the corresponding experimental results.

6.2.1 Experiment Setup

We firstly introduce the detailed information of the two real applications (spectrum and MEPS computations) and the energy related parameters. Afterwards, the off-line preparation of DOOTA algorithm is presented.

Detailed Information of Spectrum and MEPS Applications

The spectrum and MEPS applications are obtained from [100, 29, 102]: spectrum computation is used to convert signals from time domain to frequency domain and MEPS computation is adapted from Ptolemy II design environment. Figs. 6.1 and 6.2 depict the modeled DAG graphs of these two applications, respectively. Note that some of the vertexes in the DAG graphs need to be iterated q times to produce the data for the successor vertexes. Tables 6.1 and 6.2 illustrate the execution time of each vertex in spectrum and MEPS DAG graphs related to the CC2538 system-on-chip hardware, respectively. The energy consumption of sensing unit is included in vertex SRC.

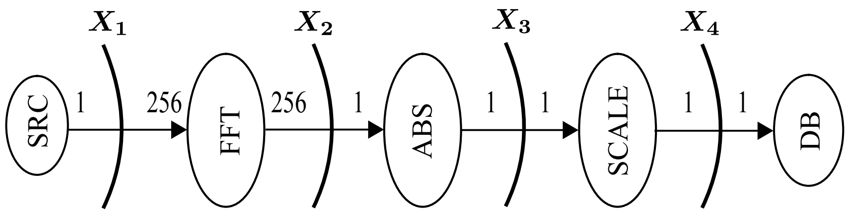


Figure 6.1: Schematic diagram of the DAG graph of spectrum computation application (modified from references [100, 29]).

The battery energy of the slave and master nodes are generated within the range of [1, 10] kJ and the energy related parameters for calculating the energy consumptions are obtained from CC2538 system-on-chip hardware as the same as illustrated in previous chapters. According to

[1] SCTTA, DCTTA, SJTA, DJTA and DTA-mhop algorithms are not analyzed, because these task allocation algorithms are extended based on CSTA, CDTA and DOOTA, and their key characteristics are similar.

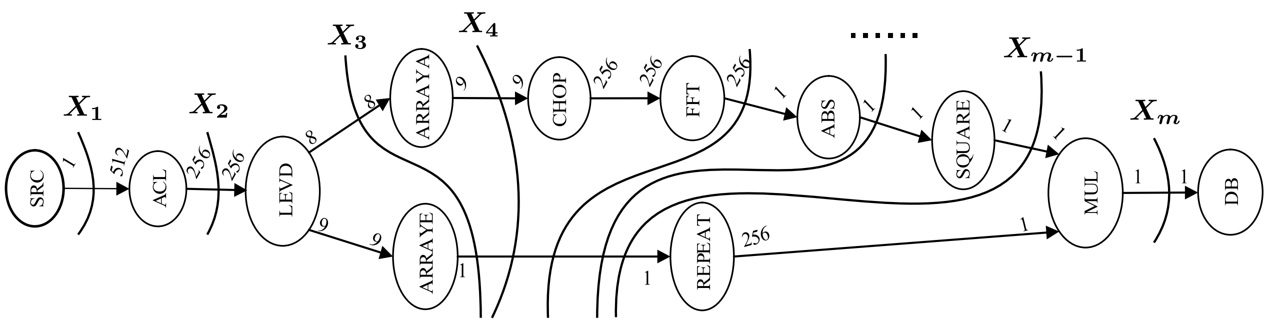


Figure 6.2: Schematic diagram of the DAG graph of MEPS computation application (modified from references [100, 29]).

Table 6.1: Execution time of each vertex in the DAG graph of spectrum application.

Actors	Average execution cycle on CC2538	Average execution time (sec.) on CC2538 with 32MHz CLK	Numbers of iterations, $q(\cdot)$
SRC	2532.99	7.92E-5	256
FFT	8163758	0.26	1
ABS	1409	4.4E-5	256
SCALE	1606.3	5.02E-5	256
DB	707	2.21E-5	256

Table 6.2: Execution time of each vertex in the DAG graph of MEPS application.

Actors	Average execution cycle on CC2538	Average execution time (sec.) on CC2538 with 32MHz CLK	Numbers of iterations, $q(\cdot)$
SRC	2534.21	7.92E-5	512
ACL	52830479	1.65	1
LEVD	17977524	0.56	1
ARRAYA	2186	6.83E-5	1
ARRAYE	325	1.01E-5	1
REPEAT	28093	8.77E-4	1
CHOP	39672	1.24E-3	1
FFT	7479003	0.23	1
ABS	1428.24	4.46E-5	256
SQUARE	478.85	1.5E-5	256
MUL	618.29	1.93E-5	256
DB	722.25	2.26E-5	256

the definition of network lifetime, i.e., the time until the first node dies, the smallest lifetime among the clusters in the network is chosen as the network lifetime.

Off-line Preparation for DOOTA Algorithm

Since each vertex (task) in the modeled DAG graph belongs to either the slave node or the master node, there are total $N = 2^K$ partition cuts of a given DAG graph with K vertexes. Taking MEPS application for example, the number of partition cuts is $N = 2^{12}$ which is a relatively large number. By using the binary decision diagram (BDD) to filter the partition cuts, 21 valid partition cuts remain. Among these cuts, there are only 3 *important partition cuts*, $\boldsymbol{X_1}$, $\boldsymbol{X_3}$ and $\boldsymbol{X_m}$ as shown in Fig. 6.2. Similarly, there are also 3 *important partition cuts* for the spectrum DAG graph. Those important partition cuts will be stored in each slave node before the deployment of the network.

6.2.2 Experimental Results

This section evaluate the proposed task allocation algorithms, CSTA, CDTA and DOOTA, by using the above introduced spectrum and MEPS applications. It firstly presents the experimental results on a general WSN scenario and then reports the evaluations of CSTA, CDTA and DOOTA within the cluster.

Evaluation on General WSNs

As shown in Fig. 6.3, a general WSN is generated with 100 nodes randomly located in a two dimensional network area of 100×100 square meters. The gateway is located at the point (50, 150) which is not shown in the figure. Using LEACH [41] clustering protocol, the 100 nodes are grouped into 6 clusters, $C_1, \cdots, C_6$, which are marked by different colors. In each cluster, the slave nodes connect with one master node by one wireless hop. Each slave node is considered to execute the same application, i.e., either spectrum computation or MEPS computation.

This section evaluates CSTA, CDTA and DOOTA algorithms on the above described scenario in terms of network lifetime increase with respect to the strategy of *no-scheduling*, in which each slave node just executes the task of the first vertex of the DAG graph. The performance is also compared with a *Heuristic* task allocation algorithm proposed by [29] that uses the same applications and the network models.

Fig. 6.4 depicts the corresponding lifetime of the 6 clusters by using different schemes. It is obvious that the task allocation algorithms extend the lifetime of each cluster much longer than the no-scheduling strategy. Among the task allocation algorithms, DOOTA performs better than

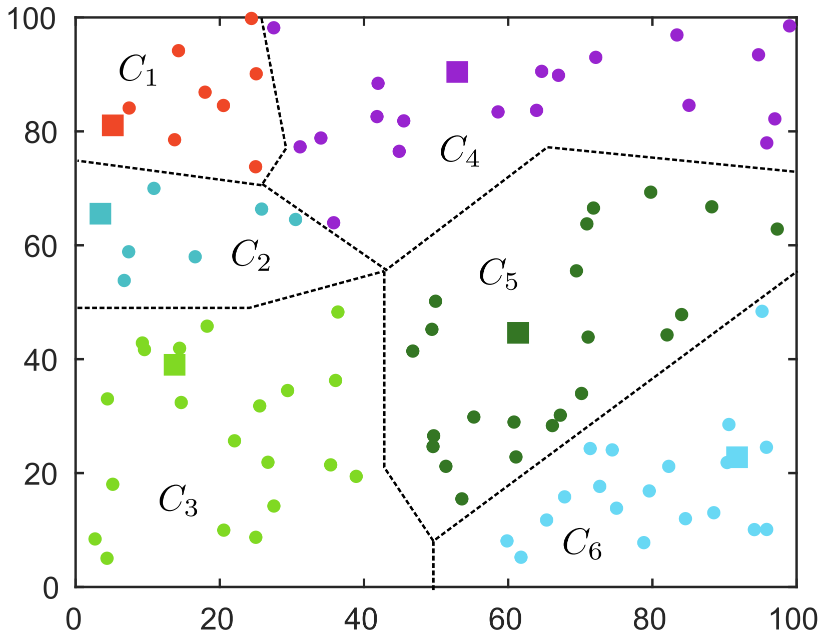
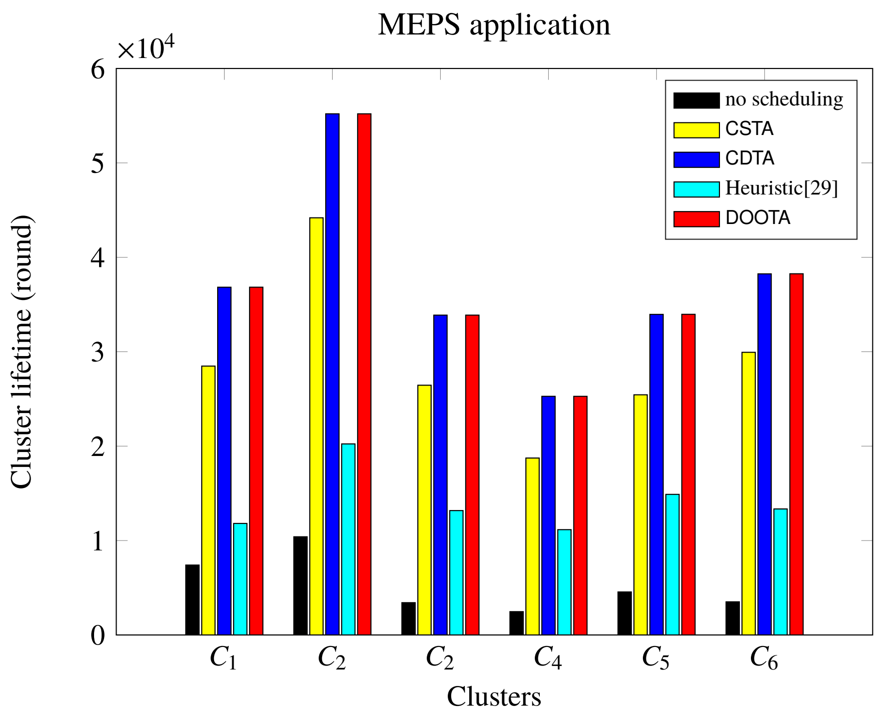


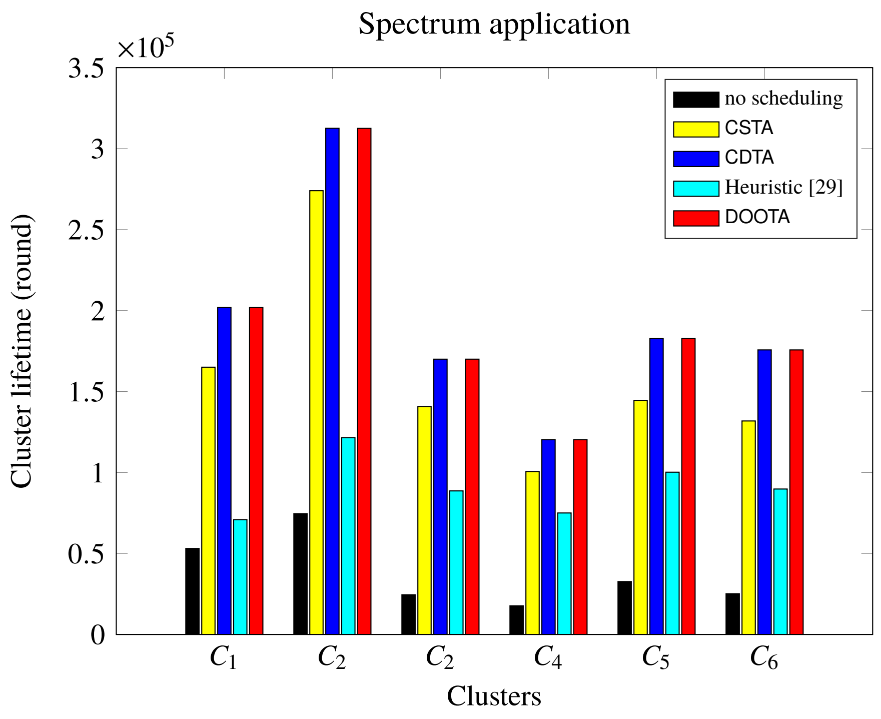
Figure 6.3: The 100-node random test WSN of size $100 \times 100\,m^2$ is grouped into 6 clusters based on the idea of LEACH [41]. The master and slave nodes are marked by ■ and • with different color in different clusters, respectively. The gateway is located at the point (50, 150) which is not shown.

CSTA and the heuristic algorithms in each cluster for both MEPS and spectrum applications. DOOTA extends the lifetime of each cluster as much as the CDTA algorithm. According to the definition of the network lifetime, the network lifetime is actually the minimum lifetime among the clusters. Taking Fig. 6.4a, the MEPS application, for example, the lifetime of cluster C_4 is considered as the network lifetime. The heuristic algorithm provides the lowest extension of the network lifetime. It considers that all slave nodes have the same battery and transmission distance to the master node. This assumption makes the heuristic scheme very simple and easy to implement, while it also brings a limitation in realistic scenarios. Rather than using the static partition solution, DOOTA and CDTA achieve the maximum network lifetime by providing different partition cuts with the corresponding weights. Specifically, both DOOTA and CDTA increase the network lifetime by 10.24 times with respect to the *no-scheduling* strategy, while the CSTA and the heuristic scheme extend the network lifetime by 7.59 and 4.52 times, respectively. The similar phenomenon holds for spectrum application as shown in Fig. 6.4b.

Although CDTA task allocation algorithm can extend the network lifetime as long as DOOTA, it is too complex and requires to know all the network parameters in advance. In drastic



(a)



(b)

Figure 6.4: The lifetimes of the 6 clusters by applying the no-scheduling strategy, CSTA, CDTA, heuristic [29] and DOOTA task allocation algorithms for cluster based WSNs when executing (a) MEPS and (b) spectrum applications.

comparison, the overhead cost of running DOOTA algorithm is so small that can be neglected as detailedly presented in the next section.

Evaluation within a Cluster

This section presents the detailed estimation of CSTA, CDTA and DOOTA algorithms within the cluster. The cluster is randomly generated in a two dimensional area of 100×100 square meters with one master located at the center and n randomly distributed slave nodes. The reported results are obtained by generating 500 instances of each experiment.

Firstly, a set of simulations are conducted to estimate the effect of the number of slave nodes, n, on the increase of network lifetime with respect to the *no-scheduling* strategy. The results, reported in Fig. 6.5, show that all of the four task allocation approaches dramatically extend the network lifetime for both MEPS and spectrum applications. As the number of slave nodes in the cluster, n, increases, their improvements become more significant. It is due to the fact that the rapid increasing workload in the cluster makes the master node overburdened and die soon, while through the efficient task allocation, the energy consumption between the slave and master nodes are well or even optimally balanced. For example, DOOTA extends the network lifetime in average from 5.90 to 20.94 times when the cluster size increases from 5 to 40 as shown in Fig 6.5a. It performs as well as CDTA, and outperforms CSTA and the heuristic [29], which is consistent with the results reported in Fig. 6.4. Moreover, the proposed task allocation algorithms have more advantages for handling complex task allocation problems. Their superiorities for executing the MEPS application (see Fig. 6.5a) is more significant than for executing the spectrum application (see Fig. 6.5b). Since there is only one master node in the cluster, executing the last task of the application is always the best partition solution as n becomes a very large number. This application-caused limitation makes the complex applications have more task allocation possibilities than the simple ones.

The next sets of simulations estimate the overhead cost of the algorithm execution. The overhead cost mainly consists of two parts: the computation and communication costs. The computation cost is measured by the execution time of running the algorithm in Matlab. Fig. 6.6 illustrates the algorithm runtime of CSTA, CDTA, DOOTA and the heuristic [29] for MEPS and spectrum applications. It is very fast to execute DOOTA and the heuristic due to their lightweight complexity, while the time consumption of CSTA and CDTA are hundreds of times higher than the execution time of DOOTA. The heuristic requires the least runtime, since it only look-ups the table, where the relation between the partition solutions and the corresponding number of slave nodes are stored off-line. Although DOOTA needs slightly more time, it provides the optimal partition solutions and achieves the maximum network lifetime. Thus, the energy cost of the

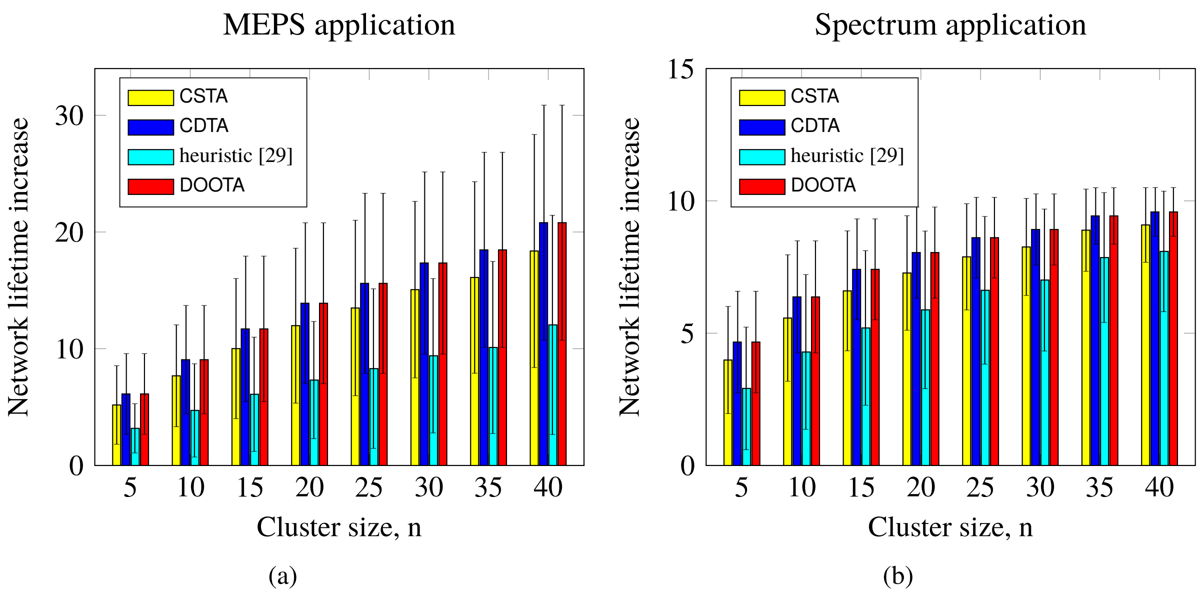


Figure 6.5: Network lifetime increase by applying the CSTA, CDTA, DOOTA and the heuristic [29] task allocation algorithms with respect to *no-scheduling* when changing the cluster size for (a) MEPS application and (b) spectrum application.

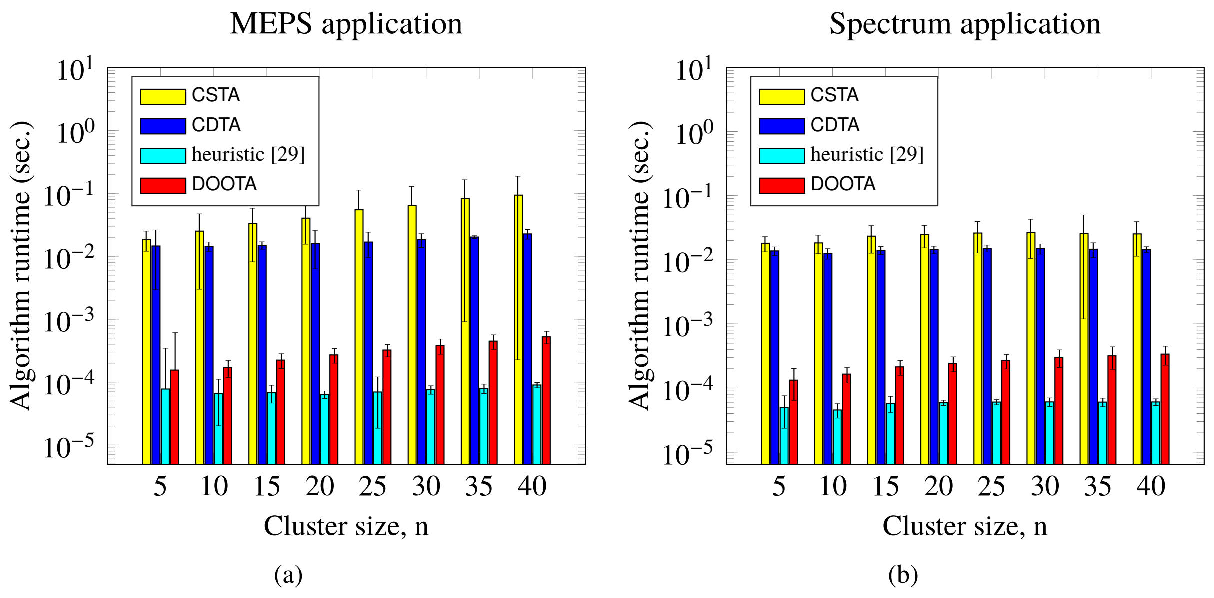


Figure 6.6: Algorithm runtime of executing the CSTA, CDTA, DOOTA and the heuristic [29] task allocation algorithms in Matlab when changing the cluster size for (a) MEPS application and (b) spectrum application.

simple computation of DOOTA can be neglected. Note that the complexities of the applications do not affect the runtime of DOOTA, which is only influenced by the number of the *important partition cuts* of the application graph. The runtime of DOOTA for both MEPS and spectrum applications are very close as shown in Fig. 6.6, since both the MEPS and spectrum application graphs have the same number of *important partition cuts* as illustrated in Section 6.2.1.

The communication overhead is measured by the number of the message exchanges between each slave and the master node. Fig. 6.7 depicts the number of the message exchanges for running the four task allocation algorithms. It can be seen that only one message exchange is needed for running CSTA, CDTA and the heuristic algorithms no matter whether it is executing MEPS or spectrum application. This is due to the fact that CSTA, CDTA and the heuristic algorithms are centralized algorithms which firstly collect the data from its slave nodes and then broadcasts the partition solutions. In contrast, running DOOTA requires a little more message exchanges. However, each node only needs in average up to 5 message exchanges as the cluster size changes from 5 to 40 for both applications. Compared with the network lifetime which lasts hundreds of thousands scheduling rounds, the overhead of executing DOOTA algorithm can be neglected. Besides, there is a slight difference between the number of message exchanges per node for MEPS and spectrum applications (see Figs. 6.7a and 6.7b), since DOOTA is operated based on the *important partition cuts* of the application graph.

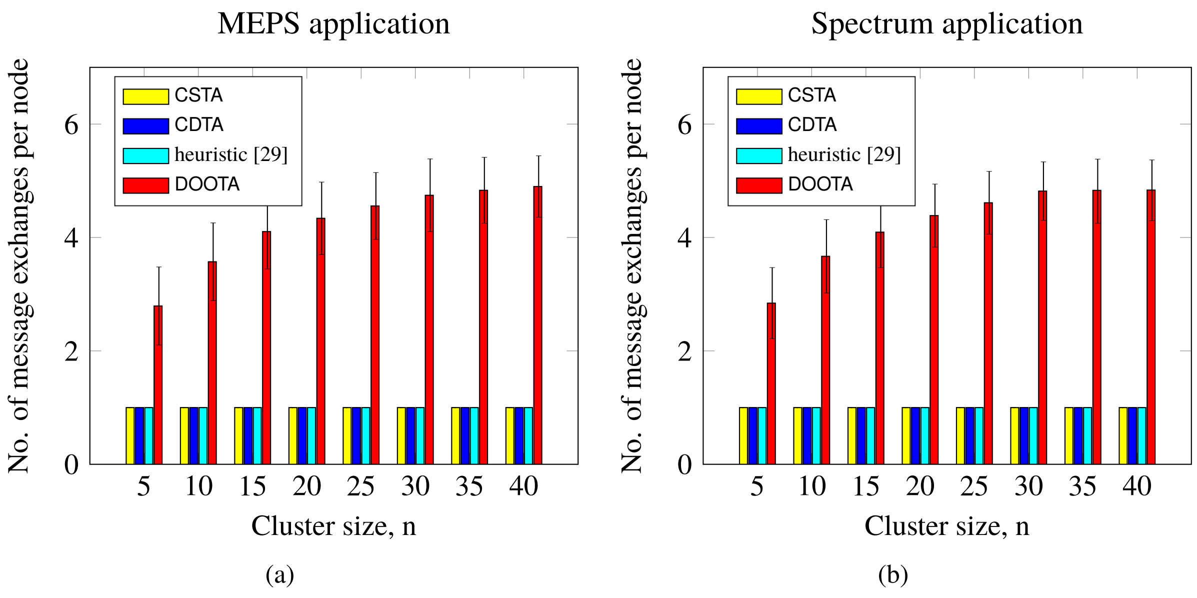


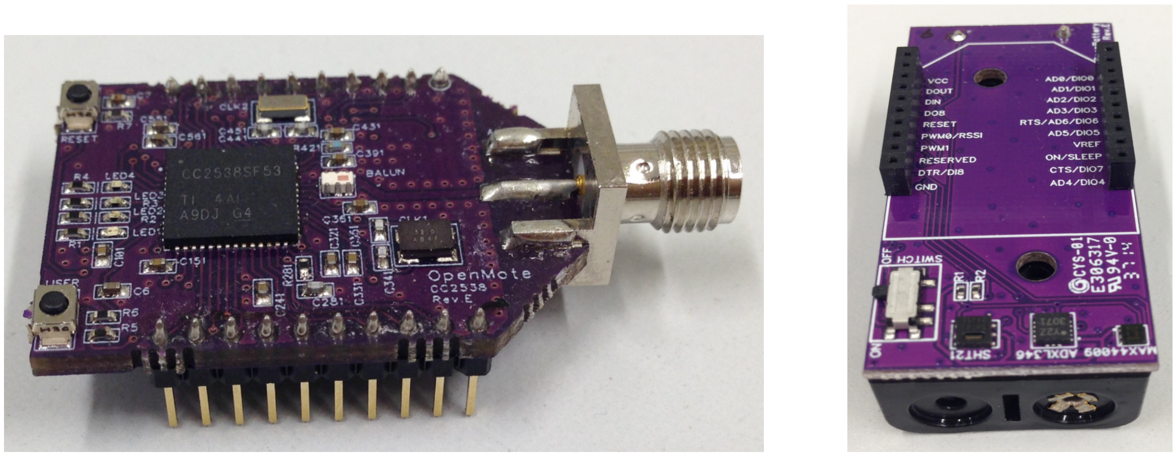
Figure 6.7: Communication overhead cost (measured by the number of message exchanges between each slave and the master node) of executing the CSTA, CDTA, DOOTA and the heuristic [29] task allocation algorithms for (a) MEPS and (b) spectrum applications.

6.3 Hardware Implementation Using OpenMote-CC2538 Platform

This section assesses the proposed CSTA, CDTA and DOOTA task allocation algorithms based on the hardware implementation using OpenMote-CC2538 platform. It firstly briefly introduces OpenMote-CC2538 and then presents the implementation results.

6.3.1 OpenMote-CC2538

In the experiments, the slave and master nodes use the same OpenMote hardware platform, which is mainly made up of a OpenMote-CC2538 connected to a OpenBattery Board, as depicted in Fig. 6.8. Note that there is also an interface board, named OpenBase, used to download and debug the code, which is not shown in this figure.



(a) (b)

Figure 6.8: OpenMote hardware: (a) OpenMote-CC2538 and (b) OpenBattery Board.

- **OpenMote-CC2538** is the core component of OpenMote hardware. It is based on the Texas Instruments CC2538 system on chip (SOC), which combines a 32-bits Cortex-M3 microcontroller (MCU) with an IEEE 802.15.4 compliant RF transceiver in one chip [96]. The MCU runs up to 32 MHz and includes a 32 Kilo bytes of RAM and a 512 Kilo bytes of flash. The RF transceiver works at the 2.4 GHz ISM band with 250 kilo bit per second (kbps) of data rate.
- **OpenBattery Board** is an interface board, which provides the power supply for OpenMote-CC2538 with two AAA batteries.

The nodes apply Contiki operating system (OS), which is an open source, highly portable and multi-tasking OS for networked embedded systems [103, 104]. They access media by using ContikiMAC [105, 106] to attain low power operation of the radio. ContikiMAC is a radio duty cycling protocol that let the nodes sleep most of the time and periodically wake up to check for radio activity using clear channel assessment(CCA). When detecting a packet transmission, the receiver stays awake to receive the next packet and then sends a link layer acknowledgment (ACK). The sender transmits the data packet until it receives the ACK. The channel check rate (CCR), specifying the number of channel checks every second, is given in powers of two. The typical settings of CCR are 2, 4, 8 (default) and 16 Hz [107]. In the experiments, the CCRs of the slave and master nodes are set to 8 and 16 Hz, respectively. Fig. 6.9 illustrates the current profile of the slave and master nodes during the communication, which is visualized on an oscilloscope by measuring the voltage drop over a fixed 10 Ω resistor. The corresponding current consumption and the duration spent on each process of the slave and master nodes during communication are listed in Table 6.3. Note that, the durations of processing data, transmitting and receiving data packets are determined by the applications.

6.3.2 Hardware Implementation Results

A symmetric WSN is formed with three slave nodes and one master node as shown in Fig. 6.10. Each slave node has to collaborate with the master node to accomplish the MEPS computation individually. All the slave and master nodes are powered by two AAA batteries, respectively, and the slave nodes use the same task allocation solutions.

The procedure of implementing CSTA and CDTA algorithms on the hardware platforms is summarized as follows. Firstly, it characterizes the energy consumption of the tasks in MEPS application on OpenMote-CC2538 platforms. Afterwards, the CSTA and CDTA algorithms are executed off-line based on the measured energy parameters as presented in Table 6.3. Then, the energy consumption of the slave and master nodes when using the obtained task allocation solutions are measured and compared with the results of no-scheduling strategy to examine the realistic network lifetime increase of CSTA and CDTA. Regarding the implementation of DOOTA algorithm, the off-line preparation procedure (as presented in Section 4.5) is firstly executed to calculate the important partition cuts of MEPS application based on the measured energy parameters. Each slave node stores these important partition cuts, based on which the DOOTA algorithm is executed among the slave and master nodes to obtain the task allocation solution. In order to estimate the optimality of DOOTA, the produced task allocation solutions are compared with the solutions provided by CSTA and CDTA. Then, they are implemented on the slave and master nodes to investigate the real gains of network lifetime improvement.

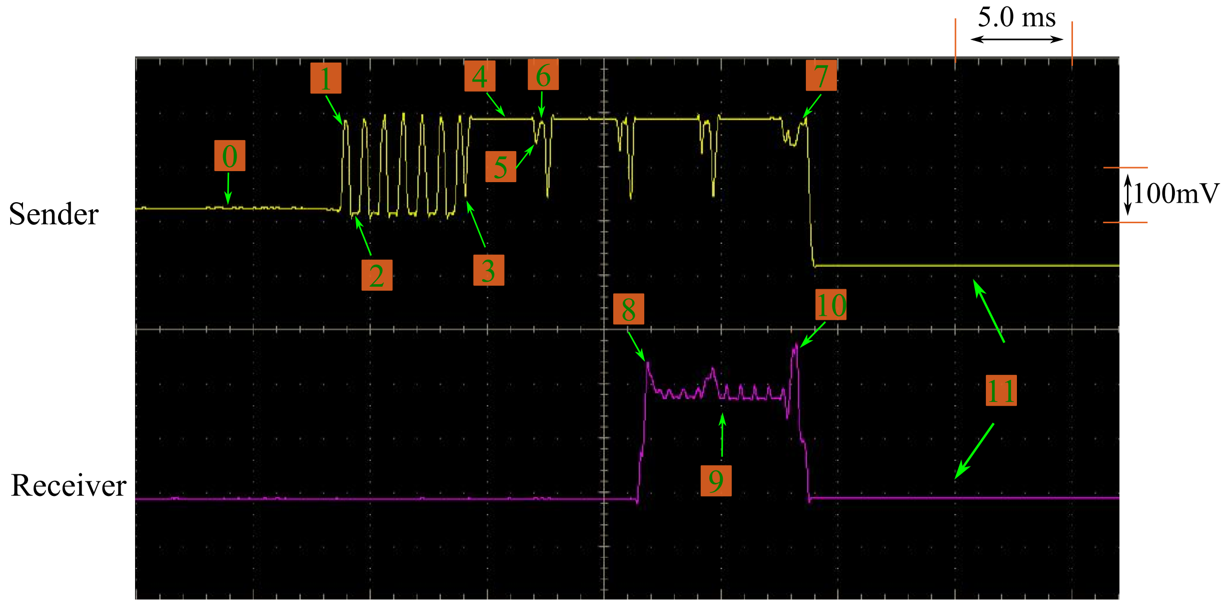


Figure 6.9: The current profile of the sender (slave node) and receiver (master node) during communication captured by oscilloscope, when the CCR of the master node is set to 16 Hz.

Table 6.3: The current consumption and the duration spent on each process of the slave and master nodes during communication, when the CCR of the master node is set to 16 Hz.

Nodes	Process	Unit Operation Description	Voltage (mV)	Current (mA)	Time (ms)	# of Units
Slave	0	Process data	118.32	11.832	Defined by APP	
	1	CCA	276.68	27.668	0.36	7
	2	CCA interval	103.76	10.376	0.34	6
	3	Switch from RX to TX	145.4	14.54	0.205	4
	4	Transmit data packet	280.84	28.084	Defined by APP	4
	5	Switch from TX to RX	230.84	23.084	0.135	4
	6	Wait to receive ACK	270.4	27.04	0.38	3
	7	Receive ACK	266.24	26.624	0.54	1
Master	8	CCA	276.68	27.668	0.36	1
	9	Receive data packet	240.74	24.074	Defined by APP	1
	10	Transmit ACK	261.74	26.174	0.54	1
	11	Low power mode		0.002		

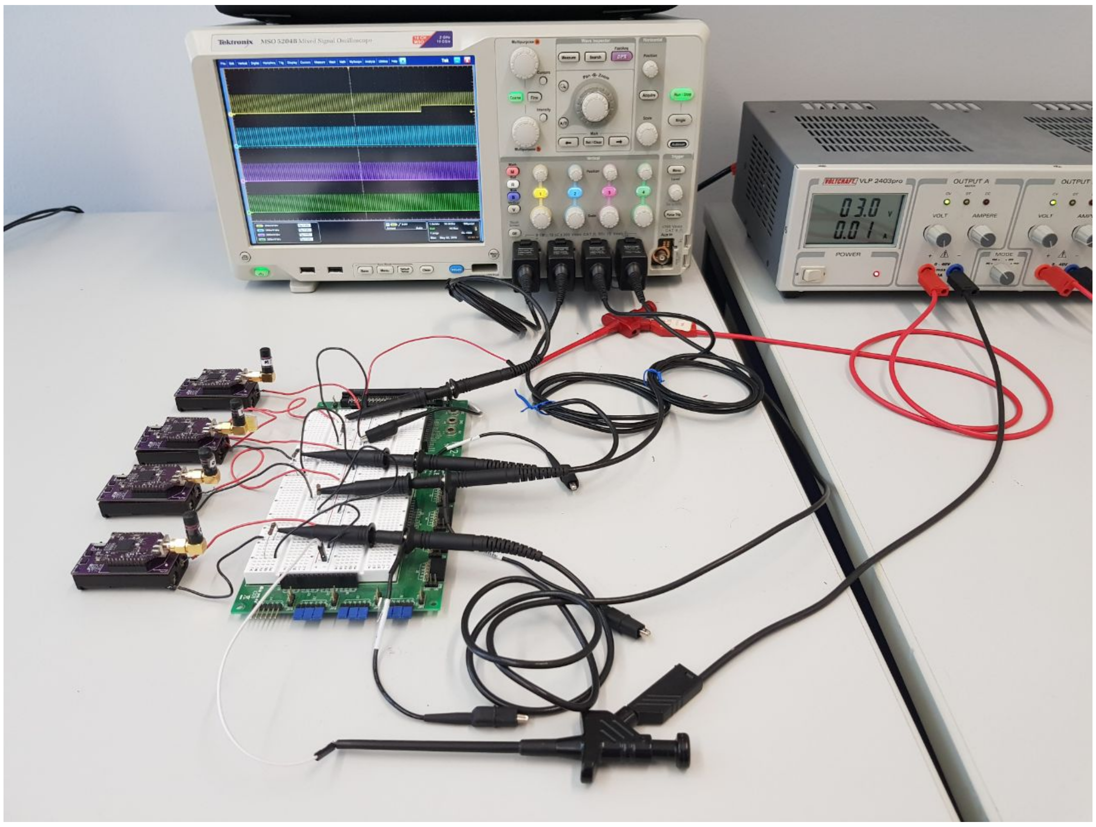


Figure 6.10: Scenario of hardware implementation.

Since the capacity of the AAA batteries, C_{bat}, is 800 mAh in the experiments, the per-time current consumption[2] is used to represent the energy consumption of the slave and master nodes. According to the parameters listed in Table 6.3, the per-time current consumption of the slave and master nodes for executing the assigned processing tasks can be expressed as $C_s = 11.832 t_s \, mAms$ and $C_m = 11.832 t_m \, mAms$, where t_s and t_m are processing durations of the slave and master nodes, respectively. Regarding the communication energy cost, the number of data packet retransmission, N_{re}, may change over time. To this end, the average value of $\overline{N}_{re}$ is used in this implementation. Thus, the per-time current consumption of the transmitting and receiving are formulated as:

$$\begin{aligned} C_{tx} &= 7C_{cca} + 6C_{cca\_itv} + \overline{N}_{re}(C_{rx2tx} + C_{t\_d} + C_{tx2rx}) + (\overline{N}_{re} - 1)C_{w\_ack} + C_{r\_ack} \\ &= 69.72 + 21.16 + (2.98 + 28.084L/250 + 3.11)\overline{N}_{re} + 10.27(\overline{N}_{re} - 1) + 14.37 \\ C_{rx} &= C_{cca} + C_{r\_d} + C_{t\_ack} \\ &= 9.96 + 24.074L/250 + 14.13 \end{aligned}$$

[2]Also known as the electric charge with the unit of coulomb C and $1C = 1A1s$.

According to [108], $\overline{N}_{re}$ is set to be 6 in the experiments.

Based on the above energy related parameters and the tasks of MEPS application as presented in Section 6.2.1, the CSTA and CDTA algorithms, are executed in Matlab, respectively. The obtained task allocation solutions by executing CSTA and CDTA are listed in Table 6.4. The corresponding expected network lifetime increase of using CSTA and CDTA algorithms with respect to no-scheduling strategy are presented in Table 6.5.

Meanwhile, the off-line preparation procedure of DOOTA algorithm is also executed in Matlab. It produces three important partition cuts, $\boldsymbol{X}_1$, $\boldsymbol{X}_3$ and $\boldsymbol{X}_m$, as illustrated in Fig. 6.2. Then, these important partition cuts are stored in each slave node. The DOOTA algorithm is executed among the slave and master nodes. It turns out that DOOTA obtains the task allocation solutions after exchanging four messages between each slave and master nodes, which is consistent with the simulation results as reported in Fig. 6.7 that the execution overhead of DOOTA is so small that can be neglected. The corresponding task allocation solutions and the expected network lifetime increase are presented in Table 6.4 and Table 6.5, respectively. It can be seen from Table 6.4 that DOOTA and CDTA provide the same task allocation solutions: partition cuts $\boldsymbol{X}_1$ and $\boldsymbol{X}_3$ with the execution probabilities of 17 % and 83 %, respectively. This observation again validates the optimality of DOOTA algorithm.

Table 6.4: Task allocation solutions provided by the no-scheduling strategy, CSTA, CDTA and DOOTA algorithms for the MEPS application (refer Fig. 6.2 to see the detailed information of $\boldsymbol{X}_1$ and $\boldsymbol{X}_3$) .

Schemes	Task allocation solution
No scheduling	$\boldsymbol{X}_1$
CSTA	$\boldsymbol{X}_3$
CDTA	17% $\boldsymbol{X}_1$ and 83% $\boldsymbol{X}_3$
DOOTA	17% $\boldsymbol{X}_1$ and 83% $\boldsymbol{X}_3$

Table 6.5: Expected network lifetime increase of using CSTA, CDTA and DOOTA algorithms with respect to no-scheduling strategy for the MEPS application.

Schemes	CSTA	CDTA	DOOTA
Expected network lifetime increase	332.43 %	395.23 %	395.23 %

Next, the task allocation solutions listed in Table 6.4 are implemented on the slave and master nodes to measure the corresponding energy consumption, respectively. The per-time current consumption of the slave and master nodes using the task allocation solutions provided by no-scheduling strategy, CSTA, CDTA and DOOTA algorithms for the MEPS computation, are illustrated in Fig. 6.11. It is obvious that CDTA and DOOTA algorithms, which provide dynamic task allocation solutions, achieve the most balanced energy consumption between the slave and master nodes. Specifically, both the differences between the energy cost of the slave and master nodes using CDTA and DOOTA are 16.718 $mAms$, while the differences between the slave and master nodes using CSTA and no-scheduling are 1626.536 $mAms$ and 8129.942 $mAms$, respectively. According to the definition of the network lifetime that the network collapses until the first node dies, the realistic improvements of network lifetime by using CSTA, CDTA and DOOTA algorithms with respect to no-scheduling strategy are listed in Table 6.6. Through the comparison between Tables 6.5 and 6.6, it can be easily obtained that the gains of CSTA, CDTA and DOOTA on hardware implementation are 323.58 %, 369.58 % and 369.58 %, respectively, which are 2.67 %, 6.49 % and 6.49 % smaller than the expected gains. These reductions are due to the fact that the ContikiMAC protocol overhead[3] of the slave and master nodes have not been considered by the optimization algorithms.

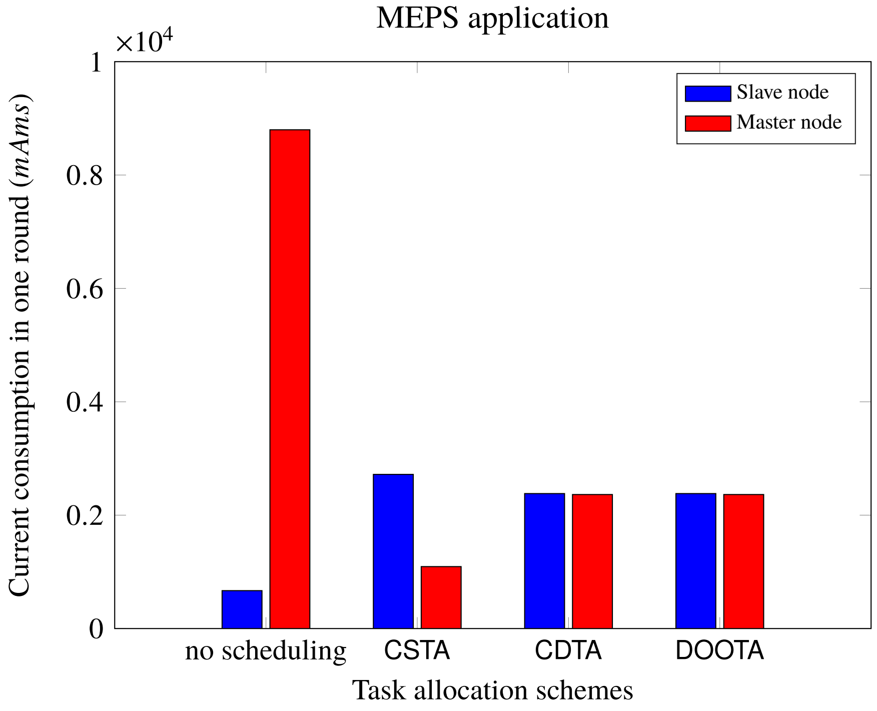


Figure 6.11: Current consumption of the slave and master nodes using the task allocation solutions provided by no-scheduling strategy, CSTA, CDTA and DOOTA algorithms for the MEPS computation in one scheduling round.

[3]This section presents the final results and the detailed description of the ContikiMAC protocol overhead of the slave and master nodes are presented in Section 8.1

Table 6.6: Realistic network lifetime increase of using CSTA, CDTA and DOOTA algorithms with respect to no-scheduling strategy for the MEPS application.

Schemes	CSTA	CDTA	DOOTA
Realistic network lifetime increase	323.58 %	369.58 %	369.58 %

6.4 Summary

This chapter has evaluated the performance of CSTA, CDTA and DOOTA algorithms for cluster based WSNs using real WSN applications and the hardware implementation based on OpenMote platform.

It firstly compares the three algorithms with a heuristic algorithm proposed by [29] using two typical applications, spectrum and MEPS computations. The experimental results show significant improvement by using the proposed task allocation algorithms. For example, both CDTA and DOOTA extend the lifetime of a general WSN by 10.24 times with respect to no-scheduling strategy for MEPS application, while CSTA and the heuristic improve the network lifetime by 7.59 and 4.52 times, respectively. Besides, the proposed task allocation algorithms are more suitable for complex applications than the simple ones, since the complex applications have more task allocation possibilities.

Then, this chapter presents the realistic gains of the proposed task allocation algorithms by hardware implementation on OpenMote-CC2538 platform. It firstly measures the energy related parameters of the OpenMote-CC2538, e.g., processing power and communication power, by connecting it with an oscilloscope. Based on these measured parameters, the CSTA, CDTA and DOOTA algorithms are executed in Matlab to calculate the corresponding task allocation solutions and the expected network lifetime increase. Afterwards, it measures the energy consumption of the slave and master nodes by applying the obtained task allocation solutions and compares with the no-scheduling strategy to examine the real network lifetime increase. The implementation results show that the realistic gains of CSTA, CDTA and DOOTA are 2.67 %, 6.49 % and 6.49 % smaller than the expected improvements, respectively. This reduction is due to the fact that the MAC protocol overhead is not considered by the optimization algorithms. Nevertheless, the proposed algorithms still significantly improve the network lifetime with respect to the no-scheduling strategy. The two dynamic algorithms, DOOTA and CDTA achieve the same network lifetime increase and they outperform the static task allocation algorithm

CSTA. Specifically, both DOOTA and CDTA extend the network lifetime 369.58 % longer than no-scheduling strategy, while CSTA achieves 323.58 % improvement.

Based on the experimental results using real applications and real hardware implementations, it can be concluded that the proposed task allocation algorithms can significantly balance the energy consumption among the nodes and dramatically prolong the network lifetime.

7 Conclusion and Outlook

7.1 Summary of Contributions **125**
7.2 Future Works **126**

Energy aware task allocation is very important for WSNs to extend the overall network lifetime by balancing the workload among the nodes. Due to the limited resources of WSN nodes and the special wireless communication, the well designed task allocation algorithms for wired systems cannot be used in WSNs. This dissertation has proposed centralized and distributed task allocation algorithms for hierarchical cluster based WSNs. Moreover, these proposed algorithms have been further extended in order to cover different task scenarios and network structures. Both the simulation and hardware implementation are conducted to evaluate the energy consumption and the network lifetime increase.

This chapter summarizes and discusses the contribution of this dissertation. In addition, it further outlines the possible extensions for future work.

7.1 Summary of Contributions

Targeting the normal applications in cluster based WSN, the task allocation problem is modeled as partitioning the application DAG graph into two subgraphs for the slave and master nodes. This work firstly proposes a centralized static task allocation algorithm (CSTA). By using a binary vector variable to represent the partition cut, CSTA formulates the task allocation problem as a binary integer linear programing (BILP) problem. It outperforms the previous heuristic algorithm and provides the optimal static partition solution which enables the slave and master nodes apply one fixed time invariant partition cut to balance their energy consumption. Secondly, motivated by the fact that using multiple partition cuts can achieve more balanced workload distribution, CSTA is extended to a centralized dynamic task allocation algorithm (CDTA). By using the average energy cost of the slave and master node in one scheduling round, CDTA uses a probability vector variable to represent the partition cuts with different weights. Based on this probability vector variable, CDTA formulates the dynamic task allocation problem as a linear programing (LP) problem. Due to the high complexity of centralized algorithms, a very lightweight distributed optimal on-line task allocation algorithm (DOOTA) is further proposed. It proves that the optimal task allocation solution is made up at most two partition cuts. Based on the extracted important partition cuts, DOOTA enables the slave and master nodes negotiate on-line to calculate the optimal task allocation solutions.

Moreover, the proposed task allocation algorithms for cluster based WSNs are further extended for different task scenarios and network structures, i.e., condition triggered applications, joint local and global applications and the multi-hop mesh network. The condition triggered applications are modeled by DAG graphs with conditional branches. The modeled conditional DAG graph is further decomposed into multiple stationary DAG graphs without conditional branches according the satisfaction probability of each condition. Based on this model, a static and a dynamic condition triggered task allocation algorithms (SCTTA and DCTTA) have been developed by formulating the conditional task allocation problem as partitioning each of the stationary DAG graph into two parts simultaneously. Further on, this work presents a static and a dynamic joint task allocation algorithms (SJTA and DJTA) for applications with both local and global tasks, respectively. The modeling of local task allocation problem does not change, while the global task allocation problem is modeled as partitioning the global DAG graph into multiple subgraphs mapping to the slave and master nodes. Besides, a dynamic task allocation algorithm for multi-hop mesh networks (DTA-mhop) is proposed in this work. The task allocation problem is modeled as partitioning the DAG graph into different subgraphs mapping to the sensor node itself, the routing nodes and the sink node. By using the summation of task allocation solutions for each sensor node, DTA-mhop formulates the task allocation problem for multi-hop network as a LP problem.

7.2 Future Works

This dissertation has proposed centralized and distributed task allocation algorithms for WSNs. However, there still exist some open issues to be solved for improving the performance of energy conservation. This section briefly summarizes these open issues as follows.

I. From a cross-layer perspective to achieve better performance, it is essential to combine the task allocation algorithms with methods in other layers, like transmission power control and adaptive radio duty cycling.

II. In order to make the task allocation algorithms applicable to the new technology scenarios, e.g., the 5th generation mobile networks (5G) and mobile edge/fog computing, it is critical to consider the application performance not only from the energy consumption point of view, but also with regards to other requirements such as bandwidth constraints, Quality of Service, and etc.

III. Moreover, reliability is another critical issue that needs to be addressed by the task allocation algorithms. For example, task assignment feedback can be introduced to detect the unsuccessful task assignment caused by wireless link errors or sensor node hardware failures and resend the task allocation solution.

8 Appendix

8.1 Current Profile of ContikiMAC Protocol Overhead

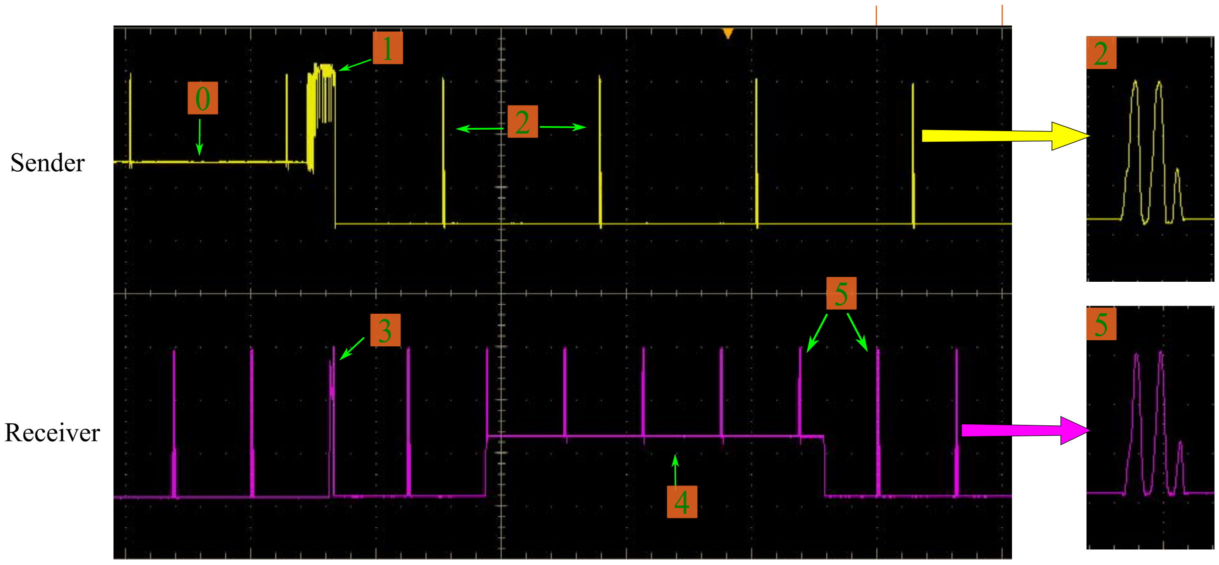


Figure 8.1: Current profile of ContikiMAC protocol overhead of the sender (slave node) and receiver (master node) captured by oscilloscope, when the CCRs of the slave and master nodes is set to 8 and 16 Hz, respectively.

Table 8.1: Descriptions of each process illustrated in Fig. 8.1

Node	Process	Operation Description
Slave	0	Processing data
	1	Transmitting
	2	CCA
Master	3	Receiving
	4	Processing data
	5	CCA

The ContikiMAC protocol overhead is mainly caused by the radio duty cycling that periodically wake up the node to check for radio activity using clear channel assessment (CCA). As shown in Fig. 8.1, process 2 and process 5 are CCAs for the slave and master nodes, respectively. The overall per-time current cost of ContikiMAC protocol overhead can be formulated as:

$$C_{mac} = C_{cca} N_{cca} f_{cca} \tag{8.1}$$

where C_{cca} is the cost for one CCA unit; N_{cca} is the number of CCA units in one instance; and f_{cca} is the wake-up frequency.

List of Notations and Acronyms

Acronym

WSN	Wireless Sensor Network.
DAG	Directed Acyclic Graph.
BILP	Binary Integer Linear Programming.
LP	Linear Programming.
IoT	Internet of Things.
MEMS	Micro-Electro-Mechanical-Systems.
ADC	Analog-to-Digital converter.
MCU	Microcontroller.
DSP	Digital Signal Processor.
ASIC	Application Specific Integrated Circuit.
FPGA	Field Programmable Gate Array.
LEACH	Low-Energy Adaptive Clustering Hierarchy.
ILP	Integer Linear Programming.
GA	Genetic Algorithm.
PSO	Particle Swarm Optimization.
TDMA	Time-Division Multiple Access.
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance.
CCA	Clear Channel Assessment.
ACK	Acknowledgment frame.
BDD	Binary Decision Diagram.
KCC	Kilo Clock Cycle.
ISM	Industrial, Scientific and Medical radio band.
MEPS	Maximum Entropy Power Spectrum.
SOC	System on Chip.
OS	Operating System.

CCR	Channel Check Rate.

Abbreviation of Proposed Approach

CSTA	Centralized Static Task Allocation algorithm for cluster based WSNs.
CDTA	Centralized Dynamic Task Allocation algorithm for cluster based WSNs.
DOOTA	Distributed On-line Optimal Task Allocation algorithm for cluster based WSNs.
SCTTA	Static Condition Triggered Task Allocation algorithm for cluster based WSNs.
DCTTA	Dynamic Condition Triggered Task Allocation algorithm for cluster based WSNs.
SJTA	Static Joint Task Allocation algorithm for cluster based WSNs.
DJTA	Dynamic Joint Task Allocation algorithm for cluster based WSNs.
DTA-mhop	Dynamic Task Allocation algorithm for Multi-HOP mesh networks.

List of Figures

2.1 Components of a general WSN node. 10

2.2 A multi-hop mesh WSN (the dotted lines represent the wireless hops). 11

2.3 A hierarchical cluster-based WSN (dotted lines represent the wireless hops among the master nodes). 12

2.4 Market-based architecture for the task allocation in WSNs (Modified from [65, 66]). 16

3.1 An example of directed acyclic graph with 5 vertexes and 5 edges. 22

3.2 An example of a DAG graph with conditional branches (there are two conditions in the application, r_1 and r_2 are the satisfaction probabilities of condition 1 and condition 2, respectively). 24

3.3 State diagram corresponding to one sender contacting one receiver without pending data in a non-beacon-enabled network. 30

3.4 State diagram corresponding to one sender transmitting data to one receiver in a beacon-enabled network. 32

4.1 Schematic diagram of task allocation for local application. 40

4.2 Schematic diagram of the dynamic task allocation using multiple partition cuts. 44

4.3 All combinations of the valid partition cuts and the corresponding energy consumption of slave node *i* and the master node. 49

4.4 BDD graph of the DAG graph in Section 4.3.1. 51

4.5 Calculation of the temporary optimal network lifetime based on the *important partition cuts*. 55

4.6 Effect of the number of slave nodes in the cluster on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms in symmetric networks, respectively (The ratio of the battery energy of the master node to the slave node is $R_{ms} = 5.0$, there are $K = 10$ tasks in the application and the middle variation level is selected, respectively). 59

4.7 Effect of the ratio of the master node's battery energy to the slave node on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms in symmetric networks, respectively (There are $n = 10$ slave node in the cluster and $K = 10$ tasks in the application, and the middle variation level is selected, respectively). 60

4.8 Effect of the number of tasks in the application on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms in symmetric networks, respectively (There are $n = 10$ slave node in the cluster, the ratio of the battery energy of the master node to the slave node is $R_{ms} = 5.0$, and the middle variation level is selected, respectively). 62

4.9 Effect of the variation among the tasks on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms in symmetric networks, respectively (There are $n = 10$ slave node in the cluster and $K = 10$ tasks in the application, and the ratio of the battery energy of the master node to the slave node is $R_{ms} = 5.0$, respectively). 63

4.10 Effect of the number of slave nodes in the cluster on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms, respectively (There are $K = 10$ tasks in the application and the middle variation level is selected, respectively). 65

4.11 Effect of the number of tasks in the application on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms, respectively (There are $n = 10$ slave node in the cluster and the middle variation level is selected, respectively). 66

4.12 Effect of the variation among the tasks on the (a) network lifetime increase and (b) algorithm runtime for CSTA, CDTA and DOOTA task allocation algorithms, respectively (There are $n = 10$ slave node in the cluster and $K = 10$ tasks in the application, respectively). 67

5.1 Schematic diagram of task allocation for condition triggered WSN application. 74

5.2 Schematic diagram of task allocation for global application. 80

5.3 A simple scenario of the multi-hop mesh WSN (the solid arrows represent the transmission directions). 87

5.4 Schematic diagram of task allocation for the multi-hop mesh network 88

5.5 The tree structure model for a general multi-hop WSN as depicted in Fig. 2.2 of Section 2.2.1 by using the minimum hop routing algorithm. 91

5.6 Effect of the number of slave nodes in the cluster on the (a) network lifetime increase and (b) algorithm runtime of SCTTA and DCTTA algorithms for the condition triggered tasks, respectively (The middle variation level among the tasks is selected and the number of conditions is 2, respectively). 95
5.7 Effect of the variation among the tasks on the (a) network lifetime increase and (b) algorithm runtime of SCTTA and DCTTA algorithms for the condition triggered tasks, respectively (There are $n = 10$ slave nodes in the cluster and the number of conditions is 2, respectively). 96
5.8 Effect of the number of conditions on the (a) network lifetime increase and (b) algorithm runtime of SCTTA and DCTTA algorithms for the condition triggered tasks, respectively (There are $n = 10$ slave nodes in the cluster and the middle level variation is selected). 97
5.9 Effect of the number of the slave nodes in the cluster on (a) network lifetime increase and (b) algorithm runtime (there are 20 and 10 tasks in global and local applications, respectively, and T = 200 rounds). 99
5.10 Effect of the number of tasks in local application on (a) network lifetime increase and (b) algorithm runtime (there are 10 slave nodes in the network and 20 tasks in global application, and T = 200 rounds). 99
5.11 Effect of the number of the tasks in global application on (a) network lifetime increase and (b) algorithm runtime (there are 10 slave nodes in the network and 10 tasks in local application, and T = 200 rounds). 101
5.12 Effect of the execution period of the global application on (a) network lifetime increase and (b) algorithm runtime (there are 10 slave nodes in the network, 20 and 10 tasks in global and local applications, respectively). 101
5.13 Effect of the number of sensor nodes on (a) network lifetime increase and (b) algorithm runtime of CDTA and the proposed DTA-mhop for multi-hop mesh WSNs (there are 10 tasks in each individual application). 103
5.14 Effect of the number of tasks in each application on (a) network lifetime increase and (b) algorithm runtime of CDTA and the proposed DTA-mhop for multi-hop mesh WSNs (there are 20 sensor nodes in the network). 104

6.1 Schematic diagram of the DAG graph of spectrum computation application (modified from references [100, 29]). 108
6.2 Schematic diagram of the DAG graph of MEPS computation application (modified from references [100, 29]). 109

6.3 The 100-node random test WSN of size $100 \times 100\, m^2$ is grouped into 6 clusters based on the idea of LEACH [41]. The master and slave nodes are marked by ■ and • with different color in different clusters, respectively. The gateway is located at the point (50, 150) which is not shown. 111
6.4 The lifetimes of the 6 clusters by applying the no-scheduling strategy, CSTA, CDTA, heuristic [29] and DOOTA task allocation algorithms for cluster based WSNs when executing (a) MEPS and (b) spectrum applications. 112
6.5 Network lifetime increase by applying the CSTA, CDTA, DOOTA and the heuristic [29] task allocation algorithms with respect to *no-scheduling* when changing the cluster size for (a) MEPS application and (b) spectrum application. 114
6.6 Algorithm runtime of executing the CSTA, CDTA, DOOTA and the heuristic [29] task allocation algorithms in Matlab when changing the cluster size for (a) MEPS application and (b) spectrum application. 114
6.7 Communication overhead cost (measured by the number of message exchanges between each slave and the master node) of executing the CSTA, CDTA, DOOTA and the heuristic [29] task allocation algorithms for (a) MEPS and (b) spectrum applications. 115
6.8 OpenMote hardware: (a) OpenMote-CC2538 and (b) OpenBattery Board. . . . 116
6.9 The current profile of the sender (slave node) and receiver (master node) during communication captured by oscilloscope, when the CCR of the master node is set to 16 Hz. 118
6.10 Scenario of hardware implementation. 119
6.11 Current consumption of the slave and master nodes using the task allocation solutions provided by no-scheduling strategy, CSTA, CDTA and DOOTA algorithms for the MEPS computation in one scheduling round. 121

8.1 Current profile of ContikiMAC protocol overhead of the sender (slave node) and receiver (master node) captured by oscilloscope, when the CCRs of the slave and master nodes is set to 8 and 16 Hz, respectively. 129

List of Tables

3.1 Seven different states of a WSN node in the communication process. 27
3.2 Parameters in the communication energy cost model of WSN nodes. 28
3.3 Values of the related parameters in the proposed communication energy cost model when applying CC2530 with IEEE 802.15.4 non-beacon-enabled mode. 31
3.4 Values of the related parameters in the proposed communication energy cost model when applying CC2430 with IEEE 802.15.4 non-beacon-enabled mode. 33

4.1 Values of the energy related parameters for calculating the energy consumption of the slave and master nodes in the networks based on CC2538 system-on-chip 57
4.2 Configuration parameters of the simulations for symmetric networks. 58

5.1 Values of the energy related parameters used in the artificially generated scenarios according to CC2538 system-on-chip datasheet. 93
5.2 Configuration parameters used in the simulations for estimation of condition triggered task allocation algorithms. 94
5.3 Configuration parameters of the simulations for estimation of the joint local and global task allocation. 98
5.4 Configuration parameters in the simulations to evaluate the performance of the task allocation algorithm for multi-hop mesh networks. 102

6.1 Execution time of each vertex in the DAG graph of spectrum application. . . . 109
6.2 Execution time of each vertex in the DAG graph of MEPS application. 109
6.3 The current consumption and the duration spent on each process of the slave and master nodes during communication, when the CCR of the master node is set to 16 Hz. 118
6.4 Task allocation solutions provided by the no-scheduling strategy, CSTA, CDTA and DOOTA algorithms for the MEPS application (refer Fig. 6.2 to see the detailed information of $\boldsymbol{X}_1$ and $\boldsymbol{X}_3$) 120
6.5 Expected network lifetime increase of using CSTA, CDTA and DOOTA algorithms with respect to no-scheduling strategy for the MEPS application. 120

6.6 Realistic network lifetime increase of using CSTA, CDTA and DOOTA algorithms with respect to no-scheduling strategy for the MEPS application. 122

8.1 Descriptions of each process illustrated in Fig. 8.1 129

Bibliography

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

[2] IEC White Paper. Internet of things: Wireless sensor networks. 2014.

[3] Mare Srbinovska, Cvetan Gavrovski, Vladimir Dimcev, Aleksandra Krkoleva, and Vesna Borozan. Environmental parameters monitoring in precision agriculture using wireless sensor networks. *Journal of Cleaner Production*, 88:297–307, 2015.

[4] Manuel Delamo, Santiago Felici-Castell, Juan J Pérez-Solano, and Andrew Foster. Designing an open source maintenance-free environmental monitoring application for wireless sensor networks. *Journal of Systems and Software*, 103:238–247, 2015.

[5] Z. Sheng, C. Mahapatra, C. Zhu, and V. C. M. Leung. Recent advances in industrial wireless sensor networks toward efficient management in iot. *IEEE Access*, 3:622–637, 2015.

[6] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen. Real-time wireless sensor-actuator networks for industrial cyber-physical systems. *Proceedings of the IEEE*, 104(5):1013–1024, May 2016.

[7] Debiao He, Neeraj Kumar, Jianhua Chen, Cheng-Chi Lee, Naveen Chilamkurti, and Seng-Soo Yeo. Robust anonymous authentication protocol for health-care applications using wireless medical sensor networks. *Multimedia Systems*, 21(1):49–60, 2015.

[8] Nicolás E Cortez, Jozué Vieira Filho, and Fabricio G Baptista. Design and implementation of wireless sensor networks for impedance-based structural health monitoring using zigbee and global system for mobile communications. *Journal of Intelligent Material Systems and Structures*, 26(10):1207–1218, 2015.

[9] Reiner Jedermann, Mike Nicometo, Ismail Uysal, and Walter Lang. Reducing food losses by intelligent food logistics. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 372(2017), 2014.

[10] Sobhi Mejjaouli and Radu F Babiceanu. Rfid-wireless sensor networks integration: Decision models and optimization of logistics systems operations. *Journal of Manufacturing Systems*, 35:234–245, 2015.

[11] N. K. Suryadevara, S. C. Mukhopadhyay, S. D. T. Kelly, and S. P. S. Gill. Wsn-based smart sensors and actuator for power management in intelligent buildings. *IEEE/ASME Transactions on Mechatronics*, 20(2):564–571, April 2015.

[12] M. Li and H. J. Lin. Design and implementation of smart home control systems based on wireless sensor networks and power line communications. *IEEE Transactions on Industrial Electronics*, 62(7):4430–4442, July 2015.

[13] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1(1):59–69, Jan 2002.

[14] V. Raghunathan, C. Schurgers, Sung Park, and M.B. Srivastava. Energy-aware wireless microsensor networks. *Signal Processing Magazine, IEEE*, 19(2):40–50, Mar 2002.

[15] Kenneth C. Barr and Krste Asanović. Energy-aware lossless data compression. *ACM Trans. Comput. Syst.*, 24(3):250–291, August 2006.

[16] Sudhanshu Tyagi and Neeraj Kumar. A systematic review on clustering and routing techniques based upon LEACH protocol for wireless sensor networks. *Journal of Network and Computer Applications*, 36(2):623 – 645, 2013.

[17] F. Ren, J. Zhang, T. He, C. Lin, and S. K. D. Ren. EBRP: Energy-balanced routing protocol for data gathering in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(12):2108–2125, Dec 2011.

[18] M. Zhao, Y. Yang, and C. Wang. Mobile data gathering with load balanced clustering and dual data uploading in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 14(4):770–785, April 2015.

[19] Yao Liang and Wei Peng. Minimizing energy consumptions in wireless sensor networks via two-modal transmission. *SIGCOMM Comput. Commun. Rev.*, 40(1):12–18, January 2010.

[20] M. Vecchio, R. Giaffreda, and F. Marcelloni. Adaptive lossless entropy compressors for tiny IoT devices. *IEEE Transactions on Wireless Communications*, 13(2):1088–1100, February 2014.

[21] W. Yu, Y. Huang, and A. Garcia-Ortiz. An altruistic compression-scheduling scheme for cluster-based wireless sensor networks. In *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 73–81, June 2015.

[22] C. Liu, K. Wu, and J. Pei. An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. *IEEE Transactions on Parallel and Distributed Systems*, 18(7):1010–1023, July 2007.

[23] Y. Huang, W. Yu, and A. Garcia-Ortiz. PKF: A communication cost reduction schema based on kalman filter and data prediction for wireless sensor networks. In *2013 IEEE International SOC Conference*, pages 73–78, Sept 2013.

[24] Y. Huang, W. Yu, C. Osewold, and A. Garcia-Ortiz. Analysis of PKF: A communication cost reduction scheme for wireless sensor networks. *IEEE Transactions on Wireless Communications*, 15(2):843–856, Feb 2016.

[25] Y. Huang, W. Yu, and A. Garcia-Ortiz. PKF-ST: A communication cost reduction scheme using spatial and temporal correlation for wireless sensor networks. In *International Conference on Embedded Wireless Systems and Networks (EWSN 2016)*, pages 47–52, Feb 2016.

[26] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson. PW-MAC: An energy-efficient predictive-wakeup mac protocol for wireless sensor networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 1305–1313, April 2011.

[27] Y. Wu, X. Y. Li, Y. Li, and W. Lou. Energy-efficient wake-up scheduling for data collection and aggregation. *IEEE Transactions on Parallel and Distributed Systems*, 21(2):275–287, Feb 2010.

[28] G. de Meulenaer, F. Gosset, F. X. Standaert, and O. Pereira. On the energy cost of communication and cryptography in wireless sensor networks. In *2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 580–585, Oct 2008.

[29] Chung-Ching Shen, William L. Plishker, Dong-Ik Ko, Shuvra S. Bhattacharyya, and Neil Goldsman. Energy-driven distribution of signal processing applications across wireless sensor networks. *ACM Trans. Sen. Netw.*, 6(3):24:1–24:32, June 2010.

[30] Yanqiu Huang Wanli Yu and Alberto Garcia-Ortiz. Energy aware task allocation in wireless sensor networks. In Habib M. Ammari, editor, *The Philosophy of Mission-Oriented Wireless Sensor Networks*. in press by Springer.

[31] Yanqiu Huang, Wanli Yu, and Alberto Garcia-Ortiz. Accurate energy-aware workload distribution for wireless sensor networks using a detailed communication energy cost model. *Journal of Low Power Electronics*, 10(2):183–193, 2014.

[32] W. Yu, Y. Huang, and A. Garcia-Ortiz. Modeling optimal dynamic scheduling for energy-aware workload distribution in wireless sensor networks. In *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 116–118, May 2016.

[33] Yanqiu Huang Wanli Yu and Alberto Garcia-Ortiz. Distributed optimal on-line task allocation algorithm for wireless sensor networks. *IEEE Sensors Journal*, 18(1):446–458, Jan 2018.

[34] W. Yu, Y. Huang, and A. Garcia-Ortiz. Joint task allocation approaches for hierarchical wireless sensor networks. In *7th International Conference on Modern Circuits and Systems Technologies (MOCAST): Electronics & Communications*, page 4, May 2018.

[35] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422, 2002.

[36] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325 – 349, 2005.

[37] Mohammad Al-Otaibi and Hamdy Soliman. Efficient geographic routeless routing protocols with enhanced location update mechanism. *International Journal of Sensor Networks*, 8(3-4):160 – 171, 2010.

[38] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados. Energy-efficient routing protocols in wireless sensor networks: A survey. *IEEE Communications Surveys Tutorials*, 15(2):551–591, Second 2013.

[39] S. s. Chiang, C. h. Huang, and K. c. Chang. A minimum hop routing protocol for home security systems using wireless sensor networks. *IEEE Transactions on Consumer Electronics*, 53(4):1483–1489, Nov 2007.

[40] Bo Wang. *Distributed Resource Allocation and Performance Optimization for Video Communication Over Mesh Networks Based on Swarm Intelligence*. PhD thesis, University of Missouri-Columbia, Dec. 2007.

[41] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10 pp. vol.2–, 2000.

[42] Anh Tuan Hoang and Mehul Motani. Collaborative broadcasting and compression in cluster-based wireless sensor networks. *ACM Trans. Sen. Netw.*, 3(3), August 2007.

[43] Priyanka Rawat, Kamal Deep Singh, Hakima Chaouchi, and Jean Marie Bonnin. Wireless sensor networks: a survey on recent developments and potential synergies. *The Journal of Supercomputing*, 68(1):1–48, Apr 2014.

[44] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292 – 2330, 2008.

[45] Ivan Stoianov, Lama Nachman, Samuel Madden, and Timur Tokmouline. PIPENET: A wireless sensor network for pipeline monitoring. *2007 6th International Symposium on Information Processing in Sensor Networks*, pages 264–273, 2007.

[46] Fatma Karray, Alberto Garcia-Ortiz, Mohamed W. Jmal, Abdulfattah M. Obeid, and Mohamed Abid. EARNPIPE: A testbed for smart water pipeline monitoring using wireless sensor network. *Procedia Computer Science*, 96:285 – 294, 2016. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 20th International Conference KES-2016.

[47] Yang Koo Lee, Young Jin Jung, and Keun Ho Ryu. *Design and Implementation of a System for Environmental Monitoring Sensor Network*, pages 223–228. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[48] H. Yang and Y. He. Wireless sensor network for orchard soil and climate monitoring. In *2009 WRI World Congress on Computer Science and Information Engineering*, volume 1, pages 58–62, March 2009.

[49] J. A. Ferre, A. Pawlowski, J. L. Guzmán, F. Rodríguez, and M. Berenguel. A wireless sensor network for greenhouse climate monitoring. In *2010 Fifth International Conference on Broadband and Biomedical Communications*, pages 1–5, Dec 2010.

[50] E. Ding, X. Liu, Y. Huang, Y. Zhang, and Wang M. Design on monitoring and measuring nodes of mine roadway separation and deformation based on wifi. *Coal Science and Technology*, 1:021, 2011.

[51] Daniel Goldsmith and James Brusey. The spanish inquisition protocol-model based transmission reduction for wireless sensor networks. In *Sensors, 2010 IEEE*, pages 2043–2048. IEEE, 2010.

[52] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco. Practical data prediction for real-world wireless sensor networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(8):2231–2244, Aug 2015.

[53] Guiyi Wei, Yun Ling, Binfeng Guo, Bin Xiao, and Athanasios V. Vasilakos. Prediction-based data aggregation in wireless sensor networks: Combining grey model and kalman filter. *Computer Communications*, 34(6):793 – 802, 2011.

[54] M. Kafil and I. Ahmad. Optimal task assignment in heterogeneous distributed computing systems. *IEEE Concurrency*, 6(3):42–50, Jul 1998.

[55] D. Bozdag, F. Ozguner, E. Ekici, and U. Catalyurek. A task duplication based scheduling algorithm using partial schedules. In *2005 International Conference on Parallel Processing (ICPP'05)*, pages 630–637, June 2005.

[56] Yang Yu and Viktor K. Prasanna. Energy-balanced task allocation for collaborative processing in wireless sensor networks. *Mob. Netw. Appl.*, 10(1-2):115–131, February 2005.

[57] C. M. Wang, C. C. Yen, W. Y. Yang, and J. S. Wang. Tree-structured linear approximation for data compression over wsns. In *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 43–51, May 2016.

[58] Zhong Shen Liang Dai, Yilin Chang. An optimal task scheduling algorithm in wireless sensor networks. *International Journal of Computers Communications & Control*, 6(1):101–112, 2011.

[59] Wanliang Wang Haiyan Shi and Ngaiming Kwok. Energy dependent divisible load theory for wireless sensor network workload allocation. *Mathematical Problems in Engineering*, 2012(235289):16, 2012.

[60] Liang Dai, Zhong Shen, Ting Chen, and Yilin Chang. Analysis and modeling of task scheduling in wireless sensor network based on divisible load theory. *International Journal of Communication Systems*, 27(5):721–731, 2014.

[61] X. Li, X. Liu, and H. Kang. Sensing workload scheduling in sensor networks using divisible load theory. In *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*, pages 785–789, Nov 2007.

[62] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, Feb 1970.

[63] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *19th Design Automation Conference*, pages 175–181, June 1982.

[64] S. Abdelhak, C. S. Gurram, S. Ghosh, and M. Bayoumi. Energy-balancing task allocation on wireless sensor networks for extending the lifetime. In *2010 53rd IEEE International Midwest Symposium on Circuits and Systems*, pages 781–784, Aug 2010.

[65] Neda Edalat, Chen-Khong Tham, and Wendong Xiao. An auction-based strategy for distributed task allocation in wireless sensor networks. *Computer Communications*, 35(8):916 – 928, 2012.

[66] N. Edalat, Wendong Xiao, C. K. Tham, E. Keikha, and Lee-Ling Ong. A price-based adaptive task allocation for wireless sensor network. In *2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*, pages 888–893, Oct 2009.

[67] J. Huang, Z. Han, M. Chiang, and H. V. Poor. Auction-based resource allocation for cooperative communications. *IEEE Journal on Selected Areas in Communications*, 26(7):1226–1237, September 2008.

[68] Ivan Mezei, Milan Lukic, Veljko Malbasa, and Ivan Stojmenovic. Auctions and imesh based task assignment in wireless sensor and actuator networks. *Computer Communications*, 36(9):979 – 987, 2013. Reactive wireless sensor networks.

[69] Y. Jin, J. Jin, A. Gluhak, K. Moessner, and M. Palaniswami. An intelligent task allocation scheme for multihop wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 23(3):444–451, March 2012.

[70] A. Gluhak Y. Jin, S. Vural and K. Moessner. Dynamic task allocation in multi-hop multimedia wireless sensor networks with low mobility. *Sensors*, 13(10):13998–14028, 2013.

[71] Z. Zeng, A. Liu, D. Li, and J. Long. A highly efficient dag task scheduling algorithm for wireless sensor networks. In *2008 The 9th International Conference for Young Computer Scientists*, pages 570–575, Nov 2008.

[72] B. Dieber, C. Micheloni, and B. Rinner. Resource-aware coverage and task assignment in visual sensor networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(10):1424–1437, Oct 2011.

[73] J. Yang, H. Zhang, Y. Ling, C. Pan, and W. Sun. Task allocation for wireless sensor network using modified binary particle swarm optimization. *IEEE Sensors Journal*, 14(3):882–892, March 2014.

[74] W. Guo, J. Li, G. Chen, Y. Niu, and C. Chen. A PSO-optimized real-time fault-tolerant task allocation algorithm in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(12):3236–3249, Dec 2015.

[75] J. Kennedy and R. C. Eberhart. A discrete binary version of the particle swarm algorithm. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 5, pages 4104–4108 vol.5, Oct 1997.

[76] J. Liu, Y. Mei, and X. Li. An analysis of the inertia weight parameter for binary particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):666–681, Oct 2016.

[77] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.

[78] Keijo Ruohonen. *Graph Theory*. Tampere University of Technology, 2013.

[79] Eduardo Casilari, Jose M Cano-García, and Gonzalo Campos-Garrido. Modeling of current consumption in 802.15. 4/zigbee sensor motes. *Sensors*, 10(6):5443–5468, 2010.

[80] M. Goyal, D. Rohm, W. Xie, S.H. Hosseini, K.S. Trivedi, Y. Bashir, and A. Divjak. A stochastic model for beaconless IEEE 802.15.4 MAC operation. *Computer Communications*, 34(12):1460 – 1474, 2011.

[81] Qin Wang, M. Hempstead, and Woodward Yang. A realistic power consumption model for wireless sensor network devices. In *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, volume 1, pages 286–295, 2006.

[82] Kim Chester. Measuring power consumption with CC2530 & Z-Stack. *Application Note AN079*, 2012.

[83] Zin Thein Kyaw and Chris Sen. Using the CC2430 and TIMAC for low-power wireless sensor applications: A power-consumption study. *Analog Applications Journal, Texas Instruments, Lit.# slyt295 Q*, 2:17–19, 2008.

[84] Texas-instruments. CC2530 datasheet. USA: Texas Instruments, Copyright, 2013.

[85] IEEE standard for information technology– local and metropolitan area networks– specific requirements– part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low rate wireless personal area networks (WPANs). *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, pages 1–320, 2006.

[86] Texas-instruments. *CC2430 Datasheet*. Chipcon Products from Texas Instruments, USA: Texas Instruments, Copyright, 2013.

[87] B Selvig. Measuring power consumption with CC2430 & Z-Stack. *Application Note AN053*, 5, 2007.

[88] Isabel Dietrich and Falko Dressler. On the lifetime of wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(1):5:1–5:39, February 2009.

[89] M. A. Abd, S. F. M. Al-Rubeaai, B. K. Singh, K. E. Tepe, and R. Benlamri. Extending wireless sensor network lifetime with global energy balance. *IEEE Sensors Journal*, 15(9):5053–5063, Sept 2015.

[90] Neda Edalat, Chen-Khong Tham, and Wendong Xiao. An auction-based strategy for distributed task allocation in wireless sensor networks. *Comput. Commun.*, 35(8):916–928, May 2012.

[91] J. Zhang, Q. Liu, and Y. Zhong. A tire pressure monitoring system based on wireless sensor networks technology. In *2008 International Conference on MultiMedia and Information Technology*, pages 602–605, Dec 2008.

[92] J.; Lv Z.; Wei W.; Song H ang, J.; Zhou. A real-time monitoring system of industry carbon monoxide based on wireless sensor networks. *Sensors*, 15(17):29535–29546, 2015.

[93] P. J. Soh, G. A. E. Vandenbosch, M. Mercuri, and D. M. M. P. Schreurs. Wearable wireless health monitoring: Current developments, challenges, and future trends. *IEEE Microwave Magazine*, 16(4):55–70, May 2015.

[94] R. Drechsler and B. Becker. *Binary decision diagrams: theory and implementation*. Springer Science & Business Media, 2013.

[95] F. Somenzi. *CUDD: CU Decision Diagram Package Release 3.0.0,*. Department of Electrical, Computer, and Energy Engineering, University of Colorado at Boulder, 2015.

[96] Texas-instruments. *CC2538 Datasheet*. Chipcon Products from Texas Instruments, USA: Texas Instruments, Copyright, 2013.

[97] Mou Wu, Liansheng Tan, and Naixue Xiong. Data prediction, compression, and recovery in clustered wireless sensor networks for environmental monitoring applications. *Information Sciences*, 329(Supplement C):800 – 818, 2016. Special issue on Discovery Science.

[98] Libin Jiang and Jean Walrand. A distributed csma algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Trans. Netw.*, 18(3):960–972, June 2010.

[99] Texas-instruments. *TMS320C5509A Datasheet*. Chipcon Products from Texas Instruments, USA: Texas Instruments, Copyright, 2017.

[100] Dong-Ik Ko, Chung-Ching Shen, ShuvraS. Bhattacharyya, and Neil Goldsman. Energy-driven partitioning of signal processing algorithms in sensor networks. In Stamatis Vassiliadis, Stephan Wong, and TimoD. Haemaelaeinen, editors, *Embedded Computer Systems: Architectures, Modeling, and Simulation*, volume 4017 of *Lecture Notes in Computer Science*, pages 142–154. Springer Berlin Heidelberg, 2006.

[101] OpenMote Technologies. Openmote features. `http://www.openmote.com/hardware/openmote-cc2538-en.html`, 2015. [Online], Accessed Nov, 30, 2016.

[102] J. Eker, J. W. Janneck, E. A. Lee, Jie Liu, Xiaojun Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Yuhong Xiong. Taming heterogeneity - the ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, Jan 2003.

[103] Thingsquare. Contiki OS homepage. `http://www.contiki-os.org/index.html`, 2015. [Online], Accessed Nov, 30, 2016.

[104] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462. IEEE, 2004.

[105] Mathieu Michel and Bruno Quoitin. Technical report: ContikiMAC vs X-MAC performance analysis. *arXiv preprint arXiv:1404.3589*, 2014.

[106] Adam Dunkels. The ContikiMAC radio duty cycling protocol. 2011.

[107] Andre Hahn Pereira. Change MAC or radio duty cycling protocols. `https://github.com/contiki-os/contiki/wiki/Change-mac-or-radio-duty-\cycling-protocols` [Online], Aug 2015. Accessed 02. Dec. 2016.

[108] Muhammad Usman. Analysis and implementation of power reduction scheme combining data compression and radio duty cycling for wireless sensor networks. Master's thesis, University of Bremen, 2017.