Wireless sensor networks in industrial environment : energy efficiency, delay and scalability

Ridha Soua

UNIVERSITY PARIS 6 PIERRE ET MARIE CURIE

PHD THESIS

to obtain the title of

DOCTEUR DE L'UNIVERSITE PIERRE ET MARIE CURIE

Specialty : Computer Science

Defended by

Ridha SOUA

Wireless Sensor Networks in Industrial Environment: Energy Efficiency, Delay and Scalability

Thesis Advisor: Pascale MINET

prepared at INRIA Paris-Rocquencourt Research center
HIPERCOM2 Team

defended on February 25, 2014

Jury :

Reviewers :	Michel Misson	-	Professor, University of Clermont I, France
	Ye-Qiong Song	-	Professor, University of Lorraine (ex-INPL), France
Advisor :	Pascale Minet	-	Research Scientist HDR, Inria Rocquencourt, France
Examiners :	Guy Pujolle	-	Professor, Pierre & Marie Curie University, France
	Stephane Lohier	-	Associate Professor, University of Paris-Est, France
	Najdib Achir	-	Associate Professor HDR, University of Paris 13
	Leila Saidane	-	Professor, University of Manouba, Tunisia
	Fabien Ligreau	-	Chargé de Développement, Safran Engineering Services

To my family who believes in me and supports me unconditionally, for their endless love and encouragement

To my uncles, aunties and cousins for their continuous support

To the soul of my grandparents who wanted to see my success

To people that I love and who love me

To everyone for being a part of my journey in the last three years

Acknowledgment

I would like to thank Professors Michel Misson and Ye-Qiong Song for serving on my reviewers. Their constructive feedback and suggestions greatly improved this thesis.

I'd like to thank all the jury members for reading my manuscript and their interest on my research. Without their help, my thesis couldn't have reached this far. So, I thank Mr. Nadjib Achir, Mr. Fabien Ligreau, Mr. Stephane Lohier and Mrs. Leila Saidane.

Definitely, road to a Phd is not always smooth but it's a pleasant journey. Many people have assisted and supported me. So I would like to express my gratitude to the people who have helped me in the accomplishment of this Ph.D. degree.

I would like to express my deeply-felt gratitude to my PhD advisor, Pascale Minet, for her warm encouragement and thoughtful guidance. The enthusiasm she has for the research was contagious and motivational to me. Her guidance taught me a lot about conducting good research. Without her support, this dissertation would not have been possible. Thank you for everything, it was a truly great experience working with you during all the periods of this journey !

Second, I am indebted to Erwan Livolant for his suggestions, comments, attentive reading, improvement of the technical quality of my work and interesting and fruitful discussions. You offered me insightful advice and unwavering support. I learned a lot from you !

I am also happy for having collaborated in OCARI Project with Cedric Adjih, Ichrak Amdouni and Tuan Dong. Thanks for your guidance and help.

I thank also Christine Anocq for her help, human support and friendly company. My administrative tasks were always easy thanks to you !

During my thesis preparation, I had a great pleasure to work with members of HIPERCOM Team. Thank you: Nadjib Ait-Saadi, Ahmed Amari, Asma Belhaoua, Philippe Jacquet, Anis Laouiti, Salman Malik, Dana Marinca, Paul Muhlethaler, Amina Naimi, Alonso Silva for your fruitful discussions and pleasant work atmosphere.

In this journey, like all journeys in life, you feel sometimes tired, exhausted or defeated. Fortunately, during my PhD journey, I have nice companions that offered me unwavering support. They are not only colleagues but very close friends: Hana Baccouch, Ichrak Amdouni, Ines ben Jemaa, Ines Khoufi, Ala-Eddine Oueslati, Ons Mabrouk, Mohamed-Haykel Zayani. I will remember the time we spent together during "pauses" :-) Thanks for all unforgettable moments...

I will forever be thankful to my former master advisor, Professor Leila Saidane. Thanks for your scientific advice and knowledge and insightful discussions and suggestions. You were and remains my model for a teacher.

Many thanks to all my colleagues and friends inside Inria and also outside for the excellent and truly enjoyable ambiance. My warmest thanks go to, hoping not to forget any of the names, Khalifa Toumi, Ali Mehdi, Chiraz Houaidia, Amine Sboui, Najmeddine Attia, Oussama Soualah, Aymen Amri, Amel Bellagoun, Fatma Cheikh, Mohamed-Said Bouksiaa, Skander Banaouas and Younes Bouchaala.

During Euromed project, I have collaborated with nice people from Tunisia, Morocco and Italia. Thank you Hanen Idoudi, Ines Elkorbi, Khawla Ghirbi, Fatma Somaa, Lucia Lo

Bello, Giuliana Alderesi, Mohamed Erradi, Abdelatif Kobbane, Mohammed-Amine Koulali, for your warmly reception and kindness.

I also had the opportunity to meet, during my master, a lot of nice people in Tunisia: Mouna Ayari, Skander Azzaz, Cherifa Boucetta, Naourez Mejri, Amine Kchich and Zied Chedhly. I am grateful to all of you, for your encouragements and the fun moments I have shared with you.

Contents

1 General Introduction **1**
- 1.1 Background and motivations . . . 1
- 1.2 Main contributions . . . 3
- 1.3 Manuscript organization . . . 4

I State of the Art **7**

2 Energy Efficient Techniques in WSNs **9**
- 2.1 Introduction . . . 9
- 2.2 Network lifetime definition . . . 10
- 2.3 Taxonomy of energy efficient techniques . . . 11
  - 2.3.1 Reasons of energy waste . . . 11
  - 2.3.2 Classification of energy efficient techniques . . . 12
- 2.4 Energy efficient routing protocols . . . 14
- 2.5 Duty cycling . . . 16
- 2.6 Conclusion . . . 18

3 Multichannel Assignment Protocols in WSNs **19**
- 3.1 Introduction . . . 20
- 3.2 Issues in multichannel communications and specificities of WSNs . . . 20
- 3.3 Particularities of channel assignment strategies in WSNs . . . 23
  - 3.3.1 Classification of mesh channel assignment strategies . . . 24
  - 3.3.2 Applicability of mesh channel assignment strategies . . . 25
- 3.4 Network architecture for multichannel communication in WSN . . . 26
  - 3.4.1 A two-level architecture . . . 26
  - 3.4.2 A three-level architecture . . . 27
  - 3.4.3 A three-level mesh architecture . . . 28
- 3.5 Network model for a multichannel WSN . . . 28
  - 3.5.1 Radio propagation . . . 29
  - 3.5.2 Interferences . . . 30

3.5.3 Connectivity . . . 31
3.6 Classification of existing multichannel assignment protocols in WSNs . . . 32
3.6.1 Categories of channel assignment method . . . 32
3.6.2 Channel selection policy . . . 35
3.6.3 Channel assignment methods . . . 36
3.6.4 Discussion . . . 38
3.7 Cross-layer design for multichannel allocation and routing . . . 40
3.8 Taxonomy proposed . . . 40
3.9 Conclusion . . . 42

II Joint Multipath Routing and Scheduling in Multichannel WSNs 43

4 Lower Bounds for Convergecast: Theoretical Study 45
4.1 Introduction and motivations . . . 45
4.2 Network models and assumptions . . . 47
4.2.1 System model . . . 47
4.2.2 Definitions . . . 48
4.2.3 Assumptions . . . 48
4.2.4 Modeling interferences for data gathering . . . 49
4.2.5 Notations . . . 51
4.3 Theoretical bounds for homogeneous traffic . . . 51
4.3.1 Linear networks . . . 52
4.3.2 Tree networks . . . 52
4.3.3 Optimal schedule . . . 52
4.4 Theoretical bounds for heterogeneous traffic . . . 56
4.4.1 Without immediate acknowledgment . . . 56
4.4.2 With immediate acknowledgment . . . 59
4.5 Conclusion . . . 63

5 MODESA: an Optimized Multichannel Slot Assignment for Raw Convergecast in WSNs 65
5.1 Introduction . . . 65
5.2 State of the art . . . 66
5.3 Multichannel slot assignment problem . . . 67
5.3.1 Formalization of the problem . . . 67
5.3.2 Illustrative examples . . . 69
5.4 MODESA: Multichannel Optimized DElay time Slot Assignment . . . 70
5.4.1 Principles . . . 70
5.4.2 The MODESA algorithm . . . 71
5.4.3 Optimality of MODESA for homogeneous traffic . . . 73
5.5 Performance evaluation of MODESA . . . 75
5.5.1 MODESA with homogeneous traffic . . . 75
5.5.2 MODESA with heterogeneous traffic . . . 80

5.5.3 Impact of additional links . . . 82
5.6 MODESA improvement . . . 84
5.6.1 Channel allocation strategy . . . 84
5.6.2 Multipath transmission scheduling . . . 86
5.6.3 Different topologies on different channels . . . 89
5.7 Conclusion . . . 92

6 MUSIKA: a MUlti-SInk Slot Assignment for Convergecast in Multichannel WSNs 93
6.1 Introduction . . . 93
6.2 State of the art . . . 94
6.3 Network model and problem formalization . . . 95
6.3.1 Network model . . . 96
6.3.2 Multi-sink multichannel convergecast problem formulation . . . 96
6.4 MUSIKA: MUlti-SInK slot Assignment . . . 98
6.4.1 Principles . . . 98
6.4.2 Algorithm . . . 99
6.5 Illustrative example . . . 101
6.6 Performance evaluation . . . 102
6.7 Conclusion . . . 104

III Adaptivity and Scalability of Joint Time Slot and Channel Assignment 105

7 An Adaptive Strategy for an Optimized Collision-Free Slot Assignment in Multichannel Wireless Sensor Networks 107
7.1 Introduction . . . 107
7.2 State of the art . . . 108
7.3 Adaptive multichannel slot assignment . . . 110
7.3.1 Definitions . . . 110
7.3.2 Assumptions . . . 111
7.3.3 Network model . . . 111
7.3.4 Theoretical bounds on the number of extra slots for a raw data convergecast . . . 113
7.3.5 AMSA: Proposed solution . . . 116
7.3.6 Illustrative example . . . 119
7.4 Performance evaluation . . . 121
7.4.1 Retransmission oriented experiments . . . 122
7.4.2 Temporary change in the application needs-oriented experiments . . . 126
7.5 Conclusion . . . 134

8 A Distributed Joint Channel and Time Slot Assignment for Convergecast in Multichannel WSNs 135
8.1 Introduction . . . 135
8.2 State of the art . . . 136
8.3 WAVE: a distributed time slot and channel assignment for convergecast in WSNs 137
8.3.1 WAVE in centralized mode . . . 137
8.3.2 WAVE in distributed mode . . . 142
8.3.3 Properties . . . 145
8.3.4 Equivalence of the two modes of WAVE . . . 146
8.3.5 Message complexity in the two modes of WAVE . . . 146
8.4 DiSCA: an optimized version of WAVE . . . 148
8.4.1 Principles . . . 149
8.4.2 DiSCA distributed algorithm . . . 150
8.5 Comparative performance evaluation of WAVE and DiSCA . . . 153
8.5.1 Homogeneous traffic . . . 153
8.5.2 Heterogeneous traffic . . . 155
8.6 Conclusion . . . 156

9 Conclusion and Perspectives 159
9.1 Synthesis . . . 159
9.2 Perspectives . . . 161

A List of Publications 165

B Résumé 167
B.1 Contexte et motivations . . . 167
B.2 Allocation conjointe de slot et canaux dans les RCsF . . . 169
B.2.1 Bornes théoriques . . . 169
B.3 MODESA: algorithme optimisé pour l'allocation conjointe de slots temporels et canaux . . . 170
B.3.1 Optimalité de MODESA . . . 170
B.3.2 Evaluation de performances . . . 171
B.4 MUSIKA: algorithme d'allocation conjointe de slots et de canaux pour les RCsF multicanaux multi-puits . . . 172
B.5 Adaptabilité et passage à l'échelle pour l'allocation conjointe des slots temporels et canaux . . . 174
B.5.1 Stratégie adaptative pour l'allocation conjointe de slot temporel et canaux 174
B.6 Allocation distribuée conjointe de slots et canaux pour les applications collecte de données . . . 176
B.6.1 WAVE: algorithme distribué d'allocation conjointe de slots temporels et canaux pour les applications de collecte de données . . . 176
B.6.2 DiSCA: Une version optimisée de WAVE . . . 177
B.6.3 Evaluation comparative de WAVE et DiSCA . . . 178
B.7 Conclusion et perspectives . . . 179

Bibliography 181

List of Figures

2.1 Taxonomy of energy efficient techniques 13

3.1 802.11 and 802.15.4 channels overlap. 21
3.2 Classification of CA approaches for Multichannel WMNs. 24
3.3 A two-level architecture for WSN. 27
3.4 A three-level architecture for WSN. 27
3.5 A three-level mesh architecture for WSN. 28
3.6 UDG and lognormal shadowing models [Ingelrest 2005] 29
3.7 Interferences in multichannel WSNs (a) primary interference (b) secondary interference. 30
3.8 Connectivity assumptions in multichannel WSNs. 32

4.1 A data gathering frame. 46
4.2 Conflicting transmissions without acknowledgment. 50
4.3 Additional conflicting transmissions with immediate acknowledgment. 50
4.4 An example of topology with 16 nodes and 2 sink interfaces where FlipFlop is not optimal. 54
4.5 Case where $\delta = 1$ when the sink is equipped with three radio interfaces and three channels. (**a**) (3 channels;3 radio interfaces) = 14 slots; (**b**) (3 channels;3 radio interfaces) = 13 slots 58
4.6 Additional conflicting transmissions with immediate acknowledgment. 60
4.7 Collision between a node and its nephew. 60
4.8 configuration whose number of slots differs with/without immediate acknowledgment. 62

5.1 The optimal number of slots, *nbOptimalSlots*, for various topologies with different numbers of sink interfaces $ninterf$ and channels *nchannel* with the notation: ($ninterf$;*nchannel*)=*nbOptimalSlots*. 70
5.2 Optimality of MODESA in T_t and T_n configurations. 75
5.3 Inaccuracy of MODESA in (a) T_t configurations (b) T_n configurations. 76

5.4 Scheduling with multiple channels: performance of MODESA and TMCP regarding the number of slots in (a) T_t configurations (b) T_n configurations. . . 77
5.5 Scheduling with multiple channels: performance of MODESA and TMCP regarding (a) max buffer (b) switch state (c) Slot reuse ratio. 77
5.6 Scheduling with multiple channels and multiple radio interfaces for sink in (a) T_t configurations (b) T_n configurations. 78
5.7 MODESA performance regarding (a) the number of required slots (b) the number of radio switches (c) the maximum buffer size (d) the throughput. . . 79
5.8 The optimal number of slots, $nbOptimalSlots$, for various topologies with different numbers of sink interfaces, $ninterf$, and channels, $n_{channel}$, with the notation: ($ninterf$;$nchannel$)=$nbOptimalSlots$. (**a**) (1;1) = 24; (**b**) (3;3) = 26; (**c**) (4;4) = 45. 80
5.9 Optimality of MODESA rin T_s and T_n configurations with heterogeneous traffic. 81
5.10 Inaccuracy of MODESA in T_s and T_n configurations with heterogeneous traffic. 81
5.11 Scheduling with multiple channels and multiple radio interfaces for sink in (a) T_t configurations (b) T_n configurations. 82
5.12 Convergecast scheduling provided by MODESA with the notation $Slot_{channel}$: (a) Schedule length of 9 slots using single channel (b) Schedule length of 11 slots when interfering links are added (c) Schedule length of 9 slots when interfering links are added and 2 channels are used. 83
5.13 Impact of additional interfering links on scheduling in (a) T_t configurations (b) T_n configurations. 84
5.14 Channel switching in case of sink equipped with (a) 1 interface and 3 channels (b) 2 interfaces and 3 channels 85
5.15 Channel load in topology with 100 nodes and sink equipped with a) 1 interface and 3 channels (b) 2 interfaces and 3 channels 85
5.16 Illustrative example of multipah routing. 88
5.17 Impact of multipath routing on Schedule length in (a) T_t configurations (b) T_n configurations 89
5.18 Impact of multipath routing on buffer size. 89
5.19 Topology diversity on different channels. 91

6.1 The two tree topologies of the network. 101
6.2 The optimal multi-sink slot assignment obtained with the GLPK solver for the illustrative example. 102
6.3 The multi-sink slot assignment obtained with MUSIKA for the illustrative example with flows f_1 and f_2 having the same priority. 102
6.4 The multi-sink slot assignment obtained with MUSIKA with flow f_2 represented in black belongs to a class of less importance degree than flow f_1 represented in white. 102
6.5 MUSIKA performances. 103

7.1 Illustrative example of the upper bound. 115
7.2 The topology of the network. 119

7.3 The primary schedule. . . . 119
7.4 The bonus slot assignment for node 6. . . . 120
7.5 The new slot assignment taking into account the additional demand of node 6. 120
7.6 The bonus slot assignment for node 9. . . . 121
7.7 The new schedule taking into account the additional demand of node 9. . . . 121
7.8 Two configurations of 20 nodes. (**a**) T_t configuration; (**b**) T_n configuration. . . . 122
7.9 Percentage of required extra slots. (**a**) in T_t configuration 7.8(a); (**b**) in T_n configuration 7.8(b). . . . 123
7.10 Percentage of required extra slots. (**a**) in T_t configuration; (**b**) in T_n configuration. . . . 124
7.11 Two configurations for the same topology of 20 nodes. (**a**) T_t configuration; (**b**) T_n configuration. . . . 125
7.12 Percentage of extra slots required. (**a**) in T_t configuration; (**b**) in T_n configuration. . . . 125
7.13 Distribution of nodes requiring extra slots. . . . 126
7.14 Impact on schedule length in case of nodes requiring extra slots. . . . 127
7.15 Average number of extra slots required with 5% of nodes having a bonus request. (**a**) in T_t configuration; (**b**) in T_n configurations. . . . 128
7.16 Average number of extra slots required with 10% of nodes having a bonus request. (**a**) in T_t configuration; (**b**) in T_n configurations. . . . 128
7.17 Average number of extra slots required with 20% of nodes having a bonus request. (**a**) in T_t configuration; (**b**) in T_n configurations. . . . 128
7.18 Slot reuse ratio with 20% of nodes having a bonus request. (**a**) in T_t configuration; (**b**) in T_n configurations. . . . 130
7.19 Average number of state switches per node with 20% of nodes having a bonus request. (**a**) in T_t configuration; (**b**) in T_n configuration. . . . 130
7.20 Sink interfaces occupation ratio with 20% of nodes having a bonus request. (**a**) in T_t configuration; (**b**) in T_n configuration. . . . 130
7.21 Distribution of nodes requiring extra slots. . . . 131
7.22 Impact of demands for bonus slots on the schedule length. . . . 132
7.23 Number of slots required for convergecast in (**a**) T_S configurations (**b**) T_N configurations. . . . 132
7.24 Optimality of MODESA recalculated and AMSA in T_t and T_n configurations. 133
7.25 Inaccuracy of MODESA recalculated and AMSA in T_t and T_n configurations. 133

8.1 Illustrative example for the $WAVE$ algorithm . . . 139
8.2 Conflicting transmissions without acknowledgment. . . . 143
8.3 Additional conflicting transmissions with immediate acknowledgment. . . . 143
8.4 Topology illustrating the difference on schedule obtained by $WAVE$ and DiSCA. 149
8.5 $WAVE$ versus optimal schedule and MODESA: homogeneous traffic . . . 153
8.6 $WAVE$ with additional interfering links . . . 154
8.7 Impact of immediate acknowledgment on schedule length provided by DiSCA 155
8.8 $WAVE$ versus optimal schedule: heterogeneous traffic . . . 155
8.9 Impact of multi-radio sink on number of slots. . . . 156

B.1 Schéma d'une supertrame. 168
B.2 Ordonnacement avec plusiers canaux: performances de MODESA et TMCP en nombre de slots dans (a) configurations T_t (b) configurations T_n. 171
B.3 Ordonnacement avec plusieurs canaux: performances de MODESA et TMCP en termes de tampons. 171
B.4 Charges sur les canaux dans des topologies de 100 noeuds et avec un puits équippé de a) 1 interface and 3 canaux (b) 2 interfaces and 3 canaux 172
B.5 Performances de MUSIKA. 173
B.6 Pourcentage du nombre de slots supplémntaires. (**a**) dans des configurations T_t; (**b**) dans des configurations T_n. 175
B.7 Nombre moyen de slots supplémentaires lorsque 20% des noeuds demandent un slot bonus (**a**) dans configurations T_t; (**b**) dans configurations T_n. 176
B.8 Comparaison du nombre de slots produit par $WAVE$, l'ordonnancement optimal et MODESA: trafic homogène 178
B.9 Nombre de slots produit par $WAVE$ avec des liens interférents additionels . . 179

List of Tables

3.1 Classification and analysis of existing multichannel assignment protocols. . . . 41

7.1 Inputs and variables of the model. 112
7.2 Extra slots scheduling when $Maxdepth = 6$ 115
7.3 Comparative table between AMSA and MODESA recalculated. 119

8.1 $WAVE$ scheduling with two channels 149
8.2 DiSCA scheduling with two channels 149

List of Algorithms

1 MODESA algorithm with the greedy variant 72
2 MUSIKA algorithm 100
3 AMSA algorithm for the sink 117
4 AMSA algorithm for any ordinary node, u 118
5 ScheduleNode Function (v, $Conflict(v)$, $nchannel$, $ScheduledNodes$) 140
6 WAVE algorithm: first wave in centralized mode 141
7 processRecSlotChAssigned Procedure ($SlotChAssigned$) 144
8 forwardRecSlotChAssigned Procedure ($SlotChAssigned$) 144
9 WAVE algorithm: first wave in distributed mode 145
10 processRecSlotChAssigned Procedure ($SlotChAssigned$) 150
11 ScheduleNode Function (v, $LastTxSlot(v)$,$EarliestPcktSlot(v)$, $Conflict(v)$, $nchannel$, $ScheduledNodes$) 151
12 DiSCA algorithm: distributed mode 152

Chapter 1

General Introduction

Contents

1.1 Background and motivations 1
1.2 Main contributions 3
1.3 Manuscript organization 4

1.1 Background and motivations

Wireless Sensor Networks (WSNs) enable the observation of the world with an unprecedented resolution. These networks are composed of many tiny low-cost low-power on-chip sensors. Typically, a sensor node includes four main components: a sensing unit for data acquisition, a microcontroller for local data processing, a communication unit to allow the transmission/reception of data to/from other connected devices and finally a small battery. Short communication ranges and limited bandwidth of sensor nodes lead to multi-hop communications and low data rates. Hence, the individual devices sense the surrounding environment and send their data, directly or via multiple hops, to a central device, namely the sink for processing. The application domain of WSNs is wide ranging from target tracking to environmental monitoring and from health monitoring to industrial applications. These application scenarios for WSNs often involve battery-powered nodes being active for a long period, without external human control after initial deployment. In the absence of energy efficient techniques, a node would drain its battery within a couple of days. This need has led researchers to design protocols able to minimize energy consumption.

Recently, WSNs have gained widespread usage in industrial environment. Indeed, current industrial applications, such as electrical power plants and aircraft control, use thousands of wired sensors in confinement conditions to measure various non critical parameters (temperature, pressure, position ...). These data are then routed to computers that process them. This leads to hundreds of kilometers of cables, a complex design and manufacturing, reliability problems particularly at connections and a large mass. For example, on an Airbus $A380$,

there are 500 kilometers of cable weighing over three tons. Resorting to wireless sensors would save hundreds of pounds while greatly reducing manufacturing complexity. In addition, in electrical power plants, radiation measurements can be delivered by sensors without compromising the life of people working in these plants. Besides, in an accident caused by an earthquake or tsunami, wired sensors networks may be damaged. However, wireless sensors can be easily deployed after plant's accident. Thus, they can provide an accurate damage assessment.

Moreover, the typical communication pattern in the industrial application is many-to-one communication. Every node plays the role of data source and/or router node through a routing tree to deliver packets to the sink. This data collection is called raw data convergecast. In this context, nodes that are near the sink should forward more packets than sensors far away. Hence, the scheduling of transmissions should be traffic-aware.

Two key issues for data convergecast raise: (1) minimized latencies and guaranteed packet delivery (2) energy saving. Minimized end-to-end delays ensure freshness of collected data. Besides, guaranteed packet delivery leads to a more accurate monitoring. Limiting factors for a fast data collection are interferences. To mitigate this problem, researchers resort to multichannel communications. Indeed, multichannel communications are exploited to increase network capacity, parallel transmissions and robustness against internal or external perturbations. Meanwhile, TelosB motes,for instance, can communicate on multiple frequencies as specified in the IEEE 802.15.4 standard. Hence, the data gathering delays can be reduced drastically.

As convergecast involves a large number of sensors that may transmit simultaneously, collisions and retransmissions represent a major challenge for bounded latencies. Indeed, collisions lead to data losses. Retransmissions increase packet latency and result on non-deterministic packet delivery times. Unlike contention-based protocols which suffer from inefficiency due to backoff and collisions, collision-free protocols guarantee bounded latencies. In fact, these protocols, also called deterministic-access protocols, ensure that any transmission of a node does not interfere with any other simultaneous transmission. It is achieved by allocation of channels and time slots to nodes in such a way that these interferences are avoided. Thus, it is easy to control the packet delay needed to reach the final destination. Furthermore, collision-free protocols are more energy efficient than contention-based protocols. They eliminate major sources of energy waste like idle listening, overhearing and collisions. In addition, a node is active only when it is transmitting to its parent or receiving from its children. Nodes turn off their radio otherwise. Therefore, collision-free protocols are ideal for limited battery powered nodes and contribute to energy saving.

Recently, the new standard $IEEE$802.15.4e proposed the TimeSlotted Channel Hopping (TSCH) [TG4e 2012] mode where nodes perform channel hopping. This latter, combined with a centrally built slotted schedule, ensures collision-free communications and high reliability against interferences. However, **the standard does not propose a mechanism to built this schedule**.

In this thesis, we focus on raw data convergecast in multichannel WSNs. We tackle the problem of finding a schedule that minimizes the delay needed to collect a large amount of data from sensors to the sink. **The aim of this work is to provide a set of conflict-**

free schedules for convergecast that orchestrate nodes activities through a shared wireless medium. These solutions should be well suited for the inherent characteristics of WSNs such as energy efficiency, delay constraints, adaptivity to environment and scalability.

1.2 Main contributions

This thesis focuses on data collection that requires timely and deterministic delivery. The intrinsic characteristics of WSNs such as limited bandwidth and scarce energy budget coupled with unreliable wireless links, channel contention and interferences, raise great challenges with regard to end-to-end delays. This thesis proposes multichannel communications to ensure the sucessful delivery of data in short delays.

The key contributions are summarized as follows:

1. **Contribution 1: Issues and specificities of multichannel communications in WSNs**
   We review the challenges raised by multichannel communications, mainly frequency of channel assignment, channel selection policy and channel assignment method. Second, we show why some channel assignments for WMNs (Wireless Mesh Networks) are not adequate for WSNs. Third, we propose WSNs architectures for multichannel communications. After having discussed the models widely used for channel assignment, we propose a classification of multichannel assignment approaches in WSNs, identifying three classes: static, dynamic and semi-dynamic approaches giving for each of them real use cases.

2. **Contribution 2: Lower bounds for raw data convergecast in multichannel WSNs**
   In some scenarios, delay constraints for data packets do not allow intermediate processing on traffic in transit, like compression or aggregation. This data collection is called raw data convergecast. We address the time-optimal convergecast problem by investigating the advantages of TDMA-based multichannel communications. We propose a very accurate definition of conflicting transmissions for data convergecast. Lower bounds for data convergecast are provided for a sink equipped with multiple transceivers and for both heterogeneous and homogeneous traffic.

3. **Contribution 3: Design and evaluation of collision-free nodes activity scheduling for multichannel WSNs**
   We design a **M**ultichannel **O**ptimized **D**Elay time Slot **A**ssignment, called MODESA. This latter is a centralized collision-free algorithm that takes advantages from multiple channels to allow parallel transmissions and improve communication reliability by avoiding noisy channels. We prove the optimality of MODESA in many multichannel topologies of WSNs. We evaluate its performances by simulations and compare it to a relevant well-known protocol designed for data convergecast in WSNs.

4. **Contribution 4: Design and evaluation of collision-free scheduling for multichannel multi-sink WSNs**

In large scale networks with a large number of sensor nodes, multiple sink nodes should be deployed, not only to increase the manageability of the network to reduce the data gathering delays or to ensure energy efficiency by balancing the load of nodes close to the sink, but also to run different functionalities of the application considered. We focus on efficient data delivery in multichannel WSNs by making use of multiple sinks. We start by giving an integer linear program of the problem that is solved by GLPK tool. Our contribution includes also, MUSIKA, a collision-free optimized time slot assignment for traffic differentiation in multi-sink WSNs.

5. **Contribution 5: Design and evaluation of an adaptive collision-free scheduling for multichannel WSNs**
   Unexpected traffic loads or retransmissions can hamper the initial time slot allocation for data convergecast in a WSN. In order to address this issue, we tackle the problem of adaptive slot assignment in multichannel WSNs. An optimization formulation using linear programming is provided and implemented using the GLPK tool. We then propose AMSA, an algorithm that unifies the management of additional slots, whatever their origin: changes in the application demands or in the medium access demands due to retransmissions.

6. **Contribution 6: Distributed joint time slot and channel assignment in multichannel WSNs**
   Sensor nodes are usually densely deployed and this huge number of sensors cannot be well handled by a centralized scheduling solution. Moreover, topology changes occur commonly in WSNs which makes re-computation of the schedule a very expensive operation. Therefore, we propose $WAVE$, a distributed scheduling for convergecast in multichannel WSNs. $WAVE$ is simple to implement, efficient and able to easily adapt to traffic changes. Extensive simulations are conducted to investigate its properties.

1.3 Manuscript organization

The rest of this manuscript is organized as follows. In Chapter 2 a small survey is presented; it covers the work done in the field of energy efficiency in WSNs. We then make a comprehensive study of multichannel assignment protocols in WSNs in Chapter 3 which corresponds to Contribution 1. The following chapters are organized into two parts:

- **Part II deals with theoretical aspect of convergecast as well centralized solution for joint time slot and channel assignment.**
  In Chapter 4, we provide lower bounds for the time needed to complete convergecast which corresponds to the Contribution 2. In Chapter 5, we focus on fast convergecast scheduling in WSNs. We explain our insights in designing MODESA a centralized multichannel joint time slot and channel assignment. Moreover, we prove its optimality in different topologies. We compare the performance of MODESA with TMCP, a well-known multichannel protocol. This corresponds to Contribution 3. Extending MODESA, we propose, MUSIKA in Chapter 6, a multichannel conflict-free schedule

that supports differentiated traffic in multi-sink WSNs. This corresponds to Contribution 4.

- **Part III tackles the adaptivity and scalability issues in raw data convergecast.**
  AMSA, an adaptive collision-free scheduling based on an incremental technique is proposed in Chapter 7 (the Contribution 5) to deal with traffic changes. The last contribution tackles the challenge of scalability in WSNs. We propse $WAVE$ a distributed joint time slot and channel assignment for convergecast.

We conclude this thesis in Chapter 9 summarizing the key results and highlighting the possible future research directions for the problems and solutions presented in the thesis.

Part I

State of the Art

Chapter 2

Energy Efficient Techniques in WSNs

Contents

2.1 Introduction 9
2.2 Network lifetime definition 10
2.3 Taxonomy of energy efficient techniques 11
2.3.1 Reasons of energy waste 11
2.3.2 Classification of energy efficient techniques 12
2.4 Energy efficient routing protocols 14
2.5 Duty cycling 16
2.6 Conclusion 18

2.1 Introduction

The myriad of potential applications supported by wireless sensor networks (WSNs) has generated much interest from the research community. Various applications range from small size low industrial monitoring to large scale energy constrained environmental monitoring. In all cases, an operational network is required to fulfill the application missions. In addition, energy consumption of nodes is a great challenge in order to maximize network lifetime. Unlike other networks, it can be hazardous, very expensive or even impossible to charge or replace exhausted batteries due to the hostile nature of environment.

Researchers are invited to design energy efficient protocols while achieving the desired network operations. This chapter focuses on different techniques to reduce the consumption of the limited energy budget of sensor nodes. After having identified the reasons of energy waste in WSNs, we classify energy efficient techniques into five classes, namely data reduction, control reduction, energy efficient routing, duty cycling and topology control. We then detail each of them, presenting subdivisions and giving many examples. We conclude by a recapitulative table.

2.2 Network lifetime definition

The most challenging concern in WSN design is how to save node energy while maintaining the desirable network behavior. Any WSN can only fulfill its mission as long as it is considered alive, but not after that. As a consequence, the goal of any energy efficient technique is to maximize network lifetime. This latter depends drastically on the lifetime of any single node. However, in the literature, there is no consensus for the definition of network lifetime. The majority of authors use a definition suitable for the context of their work. This situation has driven toward a plethora of coexisting definitions. Based on the previous works on WSNs [Minet 2009, Dietrich 2009], we give an overview of the most common definitions.

1. *Network lifetime based on the number of alive nodes*
   The definition found most frequently in the literature is the time during which all sensors are alive (also called n out of n in [Dietrich 2009], where n is the total number of sensors). The sink nodes are excluded from the set of nodes to reflect the assumption that sink nodes are more sophisticated and powerful devices. This lifetime is easy to compute since it does not take into account the topology changes. However, in dense networks where redundancy is present, this metric does not represent actually the lifetime evaluation. Therefore, the only case in which this metric can be reasonably used is if all nodes are of equal of importance and critical to network application.

   A variant defines the network lifetime as the time until the fraction of alive nodes falls below a predefined threshold β [Tian 2002]. While this definition takes redundancy into account unlike the former, it does not accurately describe the correct running of data gathering applications where the failure of at most β % of sensors near the sink can prevent the sink to receive collected data.

   In the context of clustering [Soro 2005, Blough 2002], authors define the network lifetime as the time to failure of the first cluster head. However, in most works, researchers change cluster head dynamically to balance energy consumption.

2. *Network lifetime based on coverage*
   Coverage reflects how well the network can detect an event in the monitored area. Therefore some works define the lifetime as the time during which the area of interest is covered by sensor nodes. However, even an 100% coverage is not sufficient when it does not ensure that collected data are delivered to the sink.

3. *Network lifetime based on connectivity*
   This definition is based on the ability of the network to transmit data to a sink. This definition is similar to what has been proposed in context of ad hoc networks. In [Chiasserini 2002] authors define the lifetime as the minimum time when either the percentage of alive nodes or the size of the largest connected component of the network drops below a specific threshold.

4. *Network lifetime based on application requirements*
   Some authors consider that network is alive as long as application functionalities are ensured. Kumar et al. [Kumar 2005] state "we define the lifetime of a WSN to be

the time period during which the network continually satisfies the application requirements". Tian and Georganas [Tian 2002] suggest another definition: It is the time until "the network no longer provides an acceptable event detection ratio." However, if no connectivity is guaranteed to report the event, this definition becomes irrelevant.

As a conclusion, network lifetime must take into account connectivity and coverage if needed by the application supported by WSN. Knowledge of the application requirements will enable WSN designers to refine the definition of network lifetime, leading to an evaluation more realistic and more pertinent for the application users.

2.3 Taxonomy of energy efficient techniques

We detail in this section the reasons of potential energy waste in a WSN. We then propose a taxonomy of existing energy efficient solutions, keeping in mind the resource constraint nature of sensors.

2.3.1 Reasons of energy waste

In WSNs, sensors dissipate energy while sensing, processing, transmitting or receiving data to fulfill the mission required by the application. The sensing subsystem is devoted to data acquisition. It is obvious that minimizing data generated will save energy of very constrained sensors. Redundancy inherent to WSNs will produce huge similar reporting that the network is in charge of routing to the sink. Experimental results confirm that communication subsystem is a greedy source of energy dissipation.

With regard to communication, there is also a great amount of energy wasted in states that are useless from the application point of view, such as [Minet 2009]:

- *Collision*: when a node receives more than one packet at the same time, these packets collide. All packets that cause the collision have to be discarded and the retransmission of these packets is required.
- *Overhearing*: when a sender transmits a packet, all nodes in its transmission range receive this packet even if they are not the intended destination. Thus, energy is wasted when a node receives packets that are destined to other nodes.
- *Control packet overhead*: a minimal number of control packets should be used to enable data transmissions.
- *Idle listening*: is one of the major sources of energy dissipation. It happens when a node is listening to an idle channel in order to receive possible traffic.
- *Interference*: each node located between transmission range and interference range receives a packet but cannot decode it.

As network lifetime has become the key characteristic for evaluating WSN, a panoply of techniques aimed at minimizing energy consumption and improving network lifetime, are proposed. We now give a taxonomy of these techniques.

2.3.2 Classification of energy efficient techniques

We can identify five main classes of energy efficient techniques, namely, data reduction, protocol overhead reduction, energy efficient routing, duty cycling and topology control.

1. *Data reduction*: focuses on reducing the amount of data produced, processed and transmitted. In the production step, sampling based and prediction based techniques are proposed. The former exploits the spatio-temporal correlation between samples to make data collection rate dynamic [Anastasi 2009, Willett 2004, Marbini 2003, Jain 2004]. The latter, given the past history of readings and based on the observation that sensors are capable of local computation, predicts the set of readings and so the sensing device can be turned off [Goel 2001, Gedik 2007, Goel 2006].

   In the processing and communication step, different operations on collected data have been introduced during the processing step to handle the scarcity of energy resources in a WSN. For instance, data compression [Kimura 2005] and data aggregation are examples of such techniques. For data aggregation, we can distinguish Cluster based structure [Heinzelman 2000, Heinzelman 2002, Akkaya 2005], Tree based [Zhang 2004b, Zhang 2004a] and Structure-less solutions [Fan 2006a, Fan 2006b].

2. *Protocol overhead reduction*: the aim of this technique is to increase protocol efficiency by reducing the overhead. Different techniques exist. These techniques can be subdivided into 1) adaptive transmission period depending on WSN stability or distance to the information source [Mahfoudh 2008, Pei 2000]. Indeed, communication protocols often resort to periodic message exchanges. These periodic messages are sources of overhead in WSNs 2) cross-layering with the upper and lower layers to optimize network resources while meeting application requirements [van der Schaar 2005] and 3) optimized flooding to avoid unnecessary retransmissions [Wu 1999, Dai 2004, Ingelrest 2007]. Indeed, flooding is a widely used technique in WSNs for location discovery, route establishments, querying, etc. Hence, it is a very expensive operation for battery powered sensors.

3. *Energy efficient routing*: routing protocols should be designed with the target of maximizing network lifetime by minimizing the energy consumed by the end-to-end transmission and avoiding nodes with low residual energy. Some protocols are opportunistic, taking advantage of node mobility or the broadcast nature of wireless communications to reduce the energy consumed by a transmission to the sink. Others use geographical coordinates of nodes to build a route toward the destination. Others build a hierarchy of nodes to simplify routing and reduce its overhead. Multipath routing protocols use multiple routes to achieve load balancing and robustness against routes failures. Finally, data centric protocols send data only to interested nodes in order to spare useless transmissions.

4. *Duty cycling*: duty cycling means the fraction of time nodes are active during their lifetime. The periods during which nodes sleep or are active should be coordinated and accommodated to specific applications requirements. These techniques can be further subdivided. High granularity techniques focus on selecting active nodes among all

sensors deployed in the network. Low granularity techniques deal with switching off (respectively on) the radio of active nodes when no communication is required (respectively when a communication involving this node may occur). They are highly related to the medium access protocol.

5. *Topology control*: it focuses on reducing energy consumption by adjusting transmission power while maintaining network connectivity. A new reduced topology is created based on local information.

The taxonomy of energy efficient techniques is illustrated in Figure 2.1.

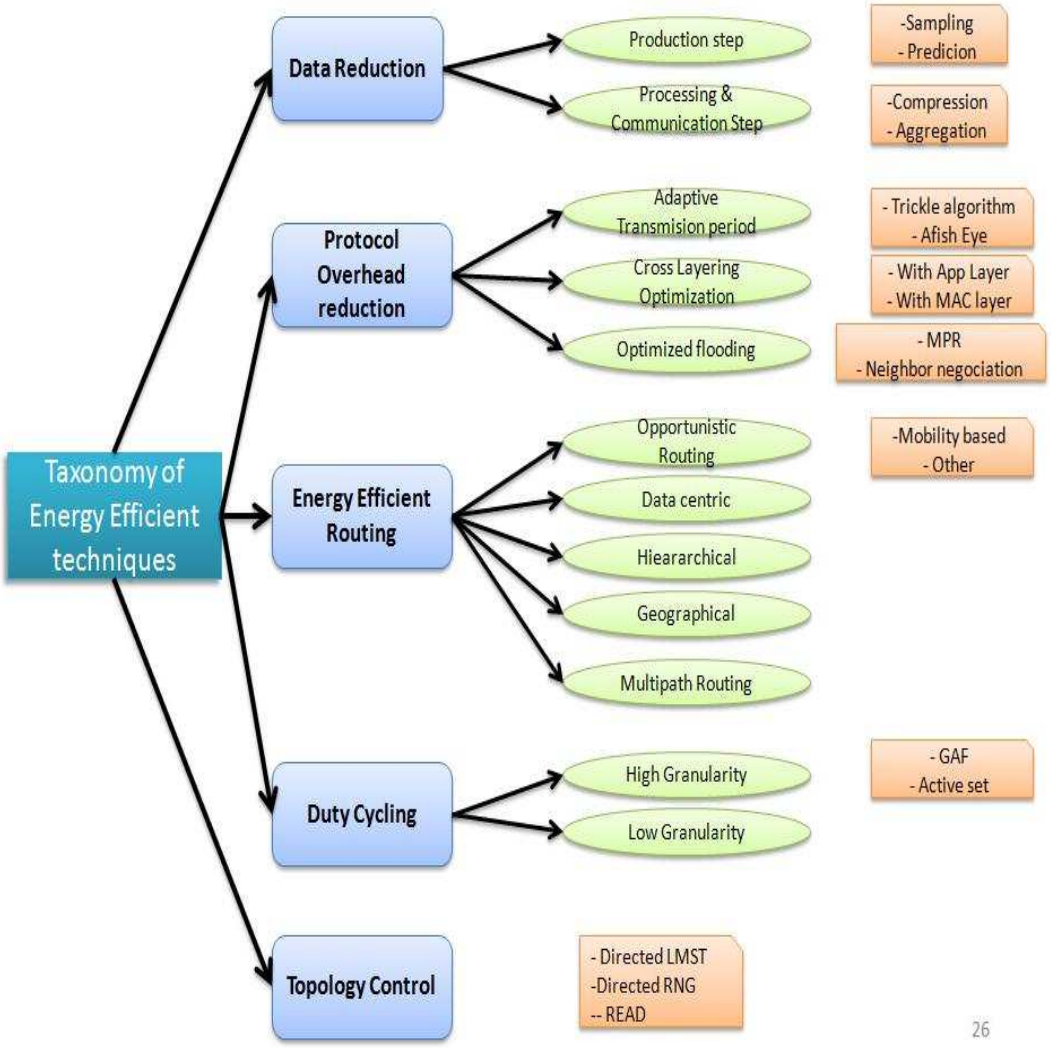


Figure 2.1: Taxonomy of energy efficient techniques

We now focus on energy efficient routing and duty cycling that are closer to our work.

2.4 Energy efficient routing protocols

The energy constraints of sensor nodes raise challenging issues on the design of routing protocols for WSNs. Proposed protocols aim at load balancing, minimizing the energy consumed by the end-to-end transmission of a packet and avoiding nodes with low residual of energy. In this section, we give a classification rather than an exhaustive list of energy efficient routing protocols. Our classification of energy efficient routing protocols generalizes the one given in [Akkaya 2005]: data centric protocols, hierarchical protocols, geographical and opportunistic protocols. We now describe each category in more details.

1. **Data centric protocols:** These protocols target energy saving by querying sensors based on their data attributes or interest. They make the assumptions that data delivery is described by a query driven model. Nodes route any data packet by looking at its content. Mainly, two approaches were proposed for interest dissemination. The first is SPIN [Blass 2008] where any node advertises the availability of data and waits for requests from interested nodes. The second is Directed Diffusion (DD) [Akkaya 2005] in which sinks broadcast an interest message to sensors, only interested nodes reply with a gradient message. Hence, both interest and gradients establish paths between sink and interested sensors. Many other proposals have being made such as rumor routing, gradient based routing, COUGAR, CADR. See [Akkaya 2005] for a comprehensive summary.

2. **Hierarchical protocols:** Clustering protocols have been developed in order to improve scalability and reduce the network traffic towards the sink. Cluster based protocols have shown lower energy consumption than flat protocols despite the overhead introduced by cluster construction and maintenance. One of the pioneering hierarchical routing protocol is LEACH [Akkaya 2005]. In this protocol, sensors organize themselves in local clusters with one node acting as a cluster head. To balance energy consumption, a randomized rotation of cluster head is used. PEGASIS is another example of hierarchical protocol [Akkaya 2005]. It enhances LEACH by organizing all nodes in a chain and letting nodes to alternate the head of the chain. TEEN is both data centric and hierarchical. It builds clusters of different levels until reaching the sink. The data centric aspect is outlined by using two thresholds for sensed attributes: hard threshold and soft threshold. The former will trigger the sensor node to transmit to its cluster head. Another transmission is only permitted when the attribute value becomes higher than the soft threshold. This mechanism can drastically reduce the number of transmissions and thus energy consumption. Since TEEN is not adaptive to periodic sensor data reporting, an extension called APTEEN [Akkaya 2005] has been proposed.

3. **Geographical protocols:** Many non geographical routing protocols suffer from scalability and efficiency restrictions because they depend on flooding for route discovery and updates. Geographical protocols take advantage of nodes location information to compute routes. In [Akkaya 2005], authors propose an energy-aware protocol called GEAR consisting of two phases. In the first phase, the message is forwarded to the

target region. In the second phase, the message is forwarded to the destination within the region.

The basic idea behind GEAR is to enhance DD by sending the interests only to a certain region rather than the whole network. GAF [Akkaya 2005] ensures energy efficiency by building virtual grids based on location information of nodes. Only a single node needs to be turned on in each cell, other nodes are kept in sleeping state. SPEED [Akkaya 2005] ensures load balancing among multiple routes with its non deterministic forwarding module.

4. **Opportunistic protocols:** The crucial idea of opportunistic routing is to exploit 1) the broadcast nature and space diversity provided by the wireless medium or 2) node mobility. We distinguish two subclasses of opportunistic routing:

   • Medium broadcast nature and space diversity based protocols:
   These techniques maintain multiple forwarding candidates and judiciously decide which sets of nodes are good and prioritized to form the forwarding candidate set. In [Zeng 2007], authors highlight how these protocols achieve better energy efficiency.
   • Mobility based protocols:
   By introducing mobility in WSN, network lifetime can be extended. Indeed, mobile nodes can move to isolated parts of the network and hence connectivity is again reached. Several works merging routing and mobility have demonstrated that this class of routing protocol exhibits smaller energy consumption when compared to classical techniques.

   - *Mobile sink based protocols*: the authors of [Luo 2005] propose a framework where mobility of the sink and routing are joint. Their proposed routing strategy offers 500 % improvement of network lifetime by using combination of sink trajectory and shortest paths. In [Bhardwaj 2002, Čagalj 2002], a learning-based approach is proposed to efficiently and reliably route data to a mobile sink. Sensors in the vicinity of the sink learn its movement pattern over time and statistically characterize it as a probability distribution function. In [Papadimitriou 2006], authors demonstrate that maximum lifetime can be achieved by solving optimally two joint problems: a scheduling problem that determines the sojourn times of the sink at different locations, and a routing problem in order to deliver the collected data to the sink in an energy efficient way.
   - *Mobile relay based protocols*: these techniques have been introduced in the context of opportunistic networks [Pelusi 2006] where the existence of an end-to-end routing path is not usually ensured. Thus, any node can be used as an intermediate hop for forwarding data closer to the destination. In [Shah 2003], authors assume the existence of mobile entities (called mules) present in the monitored area. Mules pick up data from the sensors when in close range, buffer it, and drop off the data to wired access points. Their model integrates a random walk for mobility pattern and incorporates system variables such as the number of mules, sensors and access points respectively. In [Ou 2007] data mules accommodate their trajectories for data delivery based on only local information.

5. **Multipath protocols:** Multipath routing protocols use multiple paths to forward data packets to the sink. Thus, they alleviate congestion in comparison with single-path routing protocols. For instance, CBMPR [Zhang 2007], targets low interferences. It combines cluster-based routing and multipath routing. Each path routing just passes through cluster heads. Consequently, interferences caused by intermediate nodes are avoided. RPL [Winter] is another multipath energy efficient routing protocol. The basic component of RPL is the Destination Oriented DAG which is a DAG oriented at a single sink. RPL supports only unicast traffic.

In Part II, we will see how multipath routing combined with multichannel communication can reduce end-to-end-delays and ensures load balancing.

2.5 Duty cycling

Duty cycling techniques are also called node activity scheduling techniques. They allow nodes to alternate activity and sleep periods. Indeed, only the sleep state guarantees energy saving since transmitting, receiving and idle listening consumes the scarce and expensive battery power resource. The idea is then to power off the radio subsystem each time it is possible while ensuring an operational network from the application point of view. These techniques can be applied at a high or a low granularity level. Each of them will use different means that will be briefly described.

1. **High granularity:** Generally, a large number of sensors is deployed on the monitored area. This high density leads to large redundancy. Therefore, redundant nodes should be switched off to achieve a high level of energy saving while a reduced set of nodes are kept in active mode to meet application requirements. Several works address this challenge. In [Meguerdichian 2003, Chakrabarty 2002] the selection of minimum sets of active nodes able to guarantee coverage is based on linear programming techniques. In GAF [Akkaya 2005], the monitored area is considered as a virtual grid and divided into small cells. Within each cell, only one node called the leader needs to be active and the other nodes can sleep. However, only connectivity requirements between cells are taken into account. SPAN [Chen 2002] is a connectivity driven protocol guaranteed by a coordinator eligibility criterion. Coordinators play a vital role by performing multi-hop routing while other sensors can be turned off. In [Wang 2003], the selection criteria of active nodes are based on both coverage and connectivity requirements. SPAN is enhanced by integrating a Coverage Configuration Protocol (CCP) that can provide different degrees of coverage requested by applications.
   Differently of other approaches, authors of [Cardei 2005b] divide the network nodes in disjoint sets. Each set should fulfill application requirements. At any time only one set is active while other nodes belonging to other sets can sleep. It has been proven that maximizing the number of disjoint sets is a NP-complete problem. In contrast with the work discussed above, authors of [Cardei 2005a] suggest maximizing network lifetime by dividing deployed sensor nodes into non disjoint sets.

2. **Low granularity:** This level deals with scheduling activity of nodes which have been selected as active to ensure network functionality. Even these nodes can sleep when they have no message to send or receive. Hence, node activity scheduling should be coordinated with medium access. We distinguish three classes of MAC protocols.

   - *TDMA-based*: time is divided into slots distributed among the nodes. Each slot is used to send or receive data. This technique ensures a collision free medium access to sensor nodes. It is suitable for periodic traffic. TRAMA [Rajendran 2003] is the earliest proposed traffic-adaptive TDMA-based protocol. For each time slot, one transmitter within two-hop neighbors is selected based on a distributed algorithm. Time is divided into a random access period to compete for slots and a scheduled access period. FLAMA [Rajendran 2005] is derived from TRAMA and dedicated to data gathering applications. FLAMA avoids the periodic information exchange between two-hop neighbors by transmitting upon request only. FlexiTP [Lee 2008], also proposed in the context of data gathering application, builds a data gathering tree and uses a depth first search of the tree to assign slots. Nodes can claim or remove slots based on the current information in their lookup table. A recent based TDMA protocol called TDMA-ASAP [Gobriel 2009a], proposed also in the context of data gathering application, integrates a coloring algorithm with the medium access. By allowing a node to steal an unused slot to its brother in the tree, this protocol can be adapted to various traffic conditions.
   - *Contention-based*: S-MAC [Ye 2004] tries to force neighbor nodes to adopt the same active/sleep schedule. For that purpose, neighbor nodes exchange their schedules using SYNC messages sent in the first subperiod. The second subperiod is dedicated to data exchange. However, listen and sleep periods of the protocols cannot be varied after node deployment. For this end, T-MAC [van Dam 2003] enhances S-MAC by allowing nodes to sleep again if no message has been received for a specified duration. The motivation of D-MAC [Lu 2004] is to guarantee that all nodes on a multihop path to the sink are awake when the data delivery is in progress. D-MAC schedules the active/sleep period of a node based on its depth in the forwarding tree. To reduce synchronization overhead, asynchronous sleep/wakeup schemes are based on periodic listening. In B-MAC [Polastre 2004], nodes wake up to check the channel for activity and remain active only for a short duration in the absence of traffic.
   - *Hybrid*: protocols of this category switch between TDMA and CSMA to accommodate to variable traffic patterns. The most known is Z-MAC [Rhee 2008]. It runs CSMA in low traffic and switches to TDMA in high traffic conditions. TDMA/CA [Mahfoudh 2010b] is a medium access taking advantage of node colors provided by SERENA to offer spatial reuse of the bandwidth and to minimize data delivery time to the sink in case of data gathering.

     It appears that *graph coloring* can be used to improve TDMA efficiency by allowing all nodes/edges with the same color to transmit simultaneously. We distinguish two classes of coloring: node coloring and edge coloring. While the latter assigns

time slots per link such that only the transmitter and the receiver are on, the former assigns the slot to the node which is transmitting. Centralized as well as distributed coloring algorithms exist. Some are deterministic, other resort to randomization to color the network. The smaller the number of colors, the better the coloring algorithm. In 2-hop coloring, no two nodes at one or two hops have the same color.

In order to extend the life of the WSN, energy has to be managed wisely. Node activity scheduling contributes significantly to reduce the energy consumption. Hence, it should be present in any solution that targets energy efficiency.

2.6 Conclusion

The availability of sensor devices allows a wide variety of applications to emerge. However, the resource constrained nature of sensors raises the problem of energy: how to maximize network lifetime despite a very limited energy budget? In this chapter, we have summarized different techniques that tackle the energy efficiency challenge in WSNs and classified them into five main classes as shown in Figure 2.1 that summarizes this chapter. Special attention has been devoted to node activity scheduling and energy efficient routing techniques. We will see in the rest of the manuscript how these two techniques can be combined to achieve energy efficiency.

As seen, WSNs suffer from limited channel capacity and interferences among sensor nodes or due to external sources. Many solutions have been proposed such as :

- Adaptive power control: Adjust the transmission power to a level just enough to reach the intended neighboring receiving node, resulting in a lower interference.
- Directional antennas: Concentrate the transmission power in the direction of the intended receiving node.
- Multiple channels: Transmissions in different frequency channels that do not overlap will not interfere with each other, so more transmissions can take place simultaneously without mutual interferences.

The focus of the next chapter will be the multichannel communications in WSNs. We will investigate how making use of multichannel capability helps to overcome the limitations of contention and interferences. In addition, characteristics and challenges of channel assignment protocols will be detailed.

Chapter 3

Multichannel Assignment Protocols in WSNs

Contents

3.1 Introduction 20
3.2 Issues in multichannel communications and specificities of WSNs . 20
3.3 Particularities of channel assignment strategies in WSNs 23
3.3.1 Classification of mesh channel assignment strategies 24
3.3.2 Applicability of mesh channel assignment strategies 25
3.4 Network architecture for multichannel communication in WSN . . 26
3.4.1 A two-level architecture 26
3.4.2 A three-level architecture 27
3.4.3 A three-level mesh architecture 28
3.5 Network model for a multichannel WSN 28
3.5.1 Radio propagation 29
3.5.2 Interferences 30
3.5.3 Connectivity 31
3.6 Classification of existing multichannel assignment protocols in WSNs 32
3.6.1 Categories of channel assignment method 32
3.6.2 Channel selection policy 35
3.6.3 Channel assignment methods 36
3.6.4 Discussion 38
3.7 Cross-layer design for multichannel allocation and routing 40
3.8 Taxonomy proposed 40
3.9 Conclusion 42

3.1 Introduction

With the spectacular development in radio and MEMS technologies, having sensor nodes capable of tuning efficiently their frequency over different channels is more and more straightforward. For instance, TelosB motes can communicate on multiple frequencies as specified in the 802.15.4 standard. This reality has given birth to multichannel communication paradigm in WSNs. Obviously, multichannel communication mitigates interferences, jamming and congestion which are sources of sensor's energy depletion as we have seen in the previous chapter. Whereas multichannel communications bring also challenging issues. Thus, in this chapter, we are motivated to draw a picture of multichannel assignment protocols in WSNs. After having identified the reasons of resorting to multichannel communication paradigm in WSNs and the specific issues that should be tackled, we propose a classification of multichannel assignment protocols, pointing out different channel selection policies, channel assignment categories and channel assignment methods. We conclude by a recapitulative table including many examples of existing multichannel protocols designed for WSNs.

3.2 Issues in multichannel communications and specificities of WSNs

The tremendous attraction of multichannel communication is the capacity to operate on multiple channels in order to 1) avoid interferences, specially high in dense networks or 2) increase network throughput. Multichannel communications in WSNs are challenging. On the one hand, they must be implemented in devices that have reduced resources (processing, storage, energy). On the other hand, sensor nodes are small devices whose cost per unit should remain small. For these reasons, sensor nodes are usually equipped with a single radio and simple multichannel protocols will be favored.

Compared to single channel communication, multichannel communication rises new problems or makes existing ones more complex. Some of them are shared by any wireless network:

I_1) *Multichannel deaf node*: A transmitter wrongly considers a destination node as unreachable because it does not get any response to its requests. This occurs when the destination node is tuned to another channel while the transmitter is trying to communicate with it.

I_2) *Multichannel hidden node*: In a single channel condition, a hidden node problem may occur in a configuration with at least three nodes, where at least two nodes are out of each other radio range. In a multichannel environment, the hidden node problem occurs when the node misses an RTS/CTS exchanged on one channel while listening on another, causing the hidden terminal problem despite the use of RTS/CTS signaling [So 2004b].

I_3) *Internal interferences*: Due to the broadcast nature of the wireless medium, the performance of a multichannel wireless network is drastically limited by interferences due

to concurrent transmissions on the same or adjacent channels in the same network [Raman 2009]. Within a WSN, we can distinguish two types of internal interferences:

- *inter-channel interferences:* where there exists at least one node in the WSN where two transmissions on two different (but partially overlapping) channels interfere with each other.
- *intra-channel interferences:* where there exists at least one node in the WSN where two transmissions on the same channel interfere with each other. Notice that such interferences also exist in single channel communications.

I_4) *External interferences*: 802.15.4 is not the only wireless network operating in the unlicensed band 2.4 GHz. There are WiFi, Bluetooth to name a few. Furthermore, external electromagnetic sources may create perturbations in this frequency band, like for instance electric appliance causing microwave radiation, radar polluting the 802.15.4 band. As depicted in Figure 3.1, only four channels of 802.15.4 do not overlap with WiFi channels.

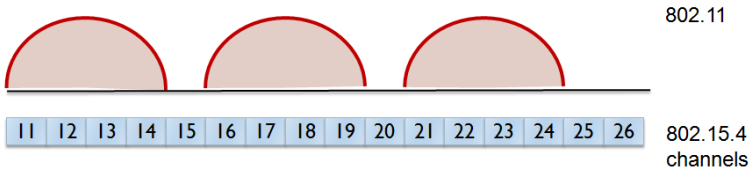


Figure 3.1: 802.11 and 802.15.4 channels overlap.

The increasing number of perturbation sources increases interferences, packet losses and retransmissions. Thus, 802.15.4 suffers from unpredictable packet delivery ratio and throughput.

Many mechanisms have been proposed to promote the coexistence between Zigbee and WiFi technologies. Authors investigate in [Liang 2010] the interferences pattern between ZigBee and WiFi networks at a bit level granularity. The experimental study distinguishes two interferences regions: 1) Symmetric region: where the front part of a 802.15.4 packet is the only part that is corrupted by interferences. 2) Asymmetric region: where any bit in the 802.15.4 packet can be corrupted: bits errors are uniformly distributed.

To circumvent interferences, authors propose headers and payloads redundancy techniques.

I_5) *Channel switching*: Whatever the channel assignment method, channel switching is performed by 1) the receiver if the channel is assigned to the sender, 2) by the sender if the channel is assigned to the receiver, or 3) by both if the channel is assigned to a link and also in case of frequency hopping. Since channel switching is not instantaneous but takes some time (around 200 μs for CC2420 radio), it can lead to packet losses within multihop flows [Incel 2011]. If each node has its own channel distinct from its

neighbors, the end-to-end transmission of a message from a sensor node to the sink requires as many channel switchings as the number of visited nodes. This may result in prohibitive delivery delays.

I_6) *Stability of links*: radio links in low-power WSNs are often unpredictable. Indeed, their quality fluctuates over time and space. Therefore, selecting high quality links is primordial for data delivery. It enhances throughput by minimizing packet losses, maximizes the network lifetime by limiting retransmissions and avoids path re-selection triggered by links failure. While the problem of links stability exists in single channel WSNs, it is more challenging in multichannel WSNs.

There are two foremost factors that lead to link unreliability [Baccour 2012]:

(a) the environment which leads to multi-path propagation effects that can differ on each channel.

(b) interferences which results from concurrent transmissions (on the same channel or on overlapping channels) within a wireless network or between coexisting wireless networks and other electromagnetic sources.

I_7) *Multicast/broadcast support*: Many multichannel assignment protocols focus on unicast transmissions enabling a sender and a receiver to tune to the same channel. However, in wireless ad hoc networks and in WSNs, broadcast communications are used to advertise a service, a gateway or more generally some information whose value has a regional scope (in between the immediate neighborhood that is restricted to one-hop neighbors and the whole network). The open question is how to support broadcast in multichannel communication? Two cases can be distinguished:
(1) all receivers are on the same channel. Hence, one copy of the message is sent by the sender.
(2) the receivers are not on the same channel. Therefore, the sender must transmit the message as many times as the number of channels used by receivers.
While the first option decreases the number of packets broadcast, the second option allows more flexibility for channel assignment.

I_8) *QoS support*: In a WSN, all messages have neither the same importance degree nor the same requirements from the application point of view. For instance, an alarm must be delivered to the sink in the shortest delays, whereas the application tolerates the loss of some sampled data as long as the number of successive losses is below a threshold. Service differentiation is required. Furthermore, the network should avoid congestion and support bursty traffic if needed.

I_9) *Autoadaptivity*: The multichannel protocol should be environment aware. Indeed, it should be able to take into account the radio environment in which the WSN operates. Channels that encounter perturbations such as jamming, external interferences or noise caused by external sources (e.g. a polluting source such as a radar) or other

coexisting wireless networks (e.g. a coexisting WiFi network) must be avoided. Usually, such channels are blacklisted for a given period. Their status should be periodically reevaluated to use them again if the perturbations have disappeared. In addition, the multichannel protocol should be able to adapt to the application requirements in order to provide the service required by the application and not more. This would result in network resources optimization.

Nevertheless, most envisioned sensor network applications encounter specific challenges that are not shared by any other wireless network:

I_{10}) *Limited capacities*: Sensor nodes are tiny devices with a small battery, limited capabilities in term of processing and memory.

I_{11}) *Low duty cycling device support*: Since most nodes are battery equipped, their energy budget is very limited. Unfortunately, this budget may also be difficult to recharge. In order to maximize network lifetime, nodes should be energy efficient. Obviously, the longer the network operates the better. To save energy, they resort to low duty cycles alternating large sleep periods and small active periods. How to support sleeping nodes?

I_{12}) *Limited bandwidth*: Wireless link bandwidth is scarce in WSNs. As sensor nodes get even tiny and WSNs grow larger in size, it is crucial to use network bandwidth efficiently and fairly. It is especially important for sensors near the sink which suffer from heavy traffic.

I_{13}) *Scalability*: WSNs may include a very large number of nodes. Multichannel communication protocols should be designed to support large and/or dense WSNs.

I_{14}) *In network processing support like data aggregation, compression*: With the ever increasing sampling rates required by the application, some processing of the sampled data could be done in the network to alleviate the amount of data transferred in the network. Many promising techniques exist such as network coding, data aggregation, data compression, data filtering.....

3.3 Particularities of channel assignment strategies in WSNs

The problem of channel assignment has been the focus of great research in Wireless Mesh Networks (WMNs). These latter consist of mesh routers and mesh clients. Fixed routers are interconnected and form a multihop backbone between the mesh clients and the Internet gateways [Si 2010]. One approach to scale the throughput is to use the multiple channels that are available in IEEE 802.11 standard. To maximize network capacity, multiple radios equip each node and can be tuned to different frequencies. That is why an efficient channel assignment is crucial for the design of WMNs. In this section, we describe the different assignment strategies in WMNs and point out why the channel allocation problem is more challenging in WSNs.

3.3.1 Classification of mesh channel assignment strategies

A recent survey [Si 2010] has classified the different channel assignment strategies in WMNs into three categories: static channel assignment, dynamic channel assignment and hybrid channel assignment. Based on this classification, Figure 3.2 illustrates the taxonomy of channel assignment in WMNs. We will use it to discuss the applicability of mesh channel assignment strategies to WSNs.

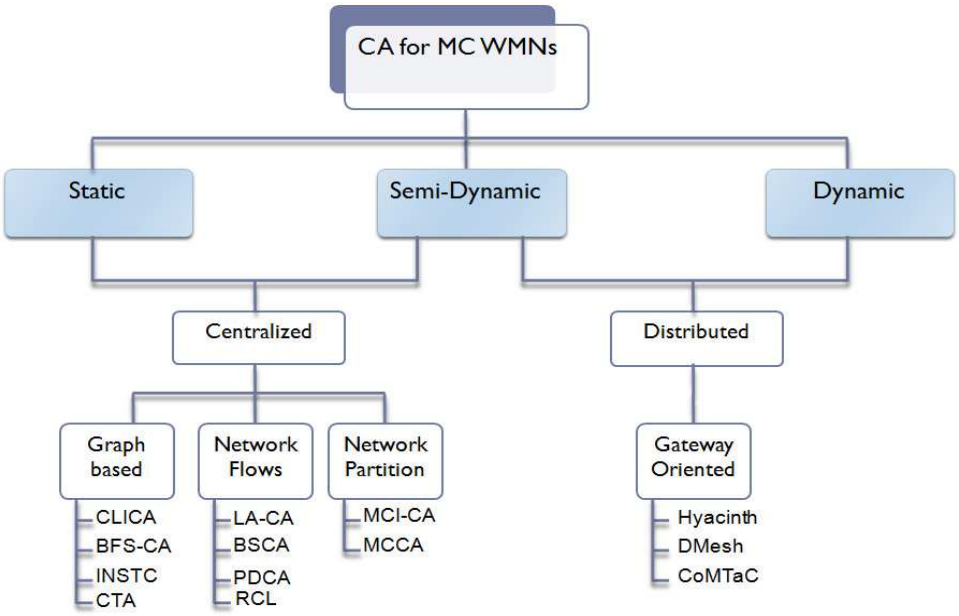


Figure 3.2: Classification of CA approaches for Multichannel WMNs.

1. *Centralized mesh channel assignment:*
   In the centralized approach, a central entity performs all needed computations. Heuristics or approximations are used to get optimal results. Therefore, all information related to network topology on each channel, conflicting nodes and tree topology should be forwarded to this entity. Hence, these solutions generally perform a static or semi-dynamic channel assignment. Some papers aim at maximizing the throughput like [Raniwala 2004, Kodialam 2005, Alicherry 2006a] or minimizing the interferences like [Marina 2005, Tang 2005, Subramanian 2008a].

   A centralized channel assignment can be envisioned for small topologies of WSNs (50 sensors) that do not exceed three hops.

2. *Distributed mesh channel assignment:*
   In distributed approaches, the channel assignment decision is distributed among all nodes. Therefore, distributed channel assignment involves communication and coordination among nodes/subtrees/ clusters. Hyacinth [Raniwala 2005b], DMesh [Das 2006] are gateway-oriented channel assignment approaches where each node is responsible for assigning channels to its children. CoMTaC [Naveed 2007a] deals with network topology changes when the channel assignment plan changes.

This category of channel assignment can be adopted in large and dense WSNs, because with multiple hops and large number of sensors, centralized solutions have scarce performances.

3.3.2 Applicability of mesh channel assignment strategies

First, we notice many similarities between WSNs and WMNs:

- The main focus of any WSN and any WMN is to create and maintain network connectivity in order to transfer the needed data from source(s) to destination node(s).
- Multi-hop networks are created. This usually requires a routing protocol.

Despite of these similarities, some proposed algorithms in [Si 2010] are not adequate to WSNs because they do not satisfy some particular issues of WSNs, like I_{10}, I_{11} and I_{13}. Hereafter, we discuss in depth the limitations of some proposed strategies in mesh network, laying down the basis of our conclusion that WSNs need new algorithms for channel assignment.

- *Hardware constraints for radio transceivers:*
  Typically, a sensor node is equipped with a half-duplex radio. Simultaneous transmissions and receptions cannot be conducted. Nevertheless, channel assignment approaches (centralized/distributed) for mesh networks assume more powerful radio hardware [Kyasanur 2006, Xing 2007, Naveed 2007b, Ko 2007]. As a consequence, many real testbeds of WMNs adopt the multi-radio multi-channel architecture [BelAirNetworks 2007, Solutions , Tropos] where nodes are capable of sensing the carrier on multiple channel simultaneously. For example, both INSTC [Tang 2005] and CLICA [Marina 2005] exhibit nodes that are equipped with multiple radios. While INSTC requires that all nodes have the same number of radios, CLICA allows a different number of radio interfaces at each node. Besides, in [Raniwala 2005b], each node has two cards (NICs): one called UP-NICs used for communicating with the parent and the second called DOWN-NICs used for communicating with the children.

  Thus, the channel assignment is more complex in WMNs than in WSNs. More precisely, the channel assignment in WMNs is usually split in two subproblems:
  (1) neighbor to interface binding: specifies through which interface a node communicates with each of its neighbors. For multichannel WSNs, this binding is envisioned for the sink only when it is equipped with multiple radios.
  (2) interface to channel binding: determines which channel is allocated to each interface. This binding remains necessary for all nodes in multichannel WSNs.

- *Overhead constraints:*
  Channel assignment in WMNs focuses on adequate mapping between the available channels and the radio at node level. Basically, the MAC layer protocols perform channel negotiation [Tzamaloukas 2001, Raniwala 2005a] using the traditional handshake mechanism. This mechanism is not suitable for WSNs which have a very limited communication bandwidth. In addition, WSNs packets are very small (packet size is between 30-50

bytes) compared to mesh packet size. RTS/CTS control packets yield to an overhead that cannot be supported by WSNs. Furthermore, the authors of [Zhou 2007] show that the exposed/hidden terminal problem is not handled properly in multichannel context. They propose a dual busy tone technique ensuring collision-free communication. However, this technique requires that any sensor mote is equipped with several transceivers. So, any node can simultaneously support a transmission on the first channel and simultaneous receptions on another one [Ruzzelli 2006]. This assumption is in contradiction with the target to keep hardware WSNs platforms relatively simple and inexpensive.

- *Processing capacity constraints:*
  The channel to radio interface assignment can be considered as an optimization problem [Shin 2012, Subramanian 2008b]. In [Shin 2012], authors propose a distributed online algorithm for the optimal channel assignment by solving Linear Programming (LP) relaxation. The joint channel assignment and routing in [Alicherry 2006b] describes a centralized solution through LP relaxation of the underlying mixed integer linear program.

  While these proposals aim to solve the issue of channel allocation in mesh networks, they also exhibit significant processing capacity. This latter is not challenging in mesh networks whereas it represents an obstacle to the use of such proposals in WSNs. Indeed, sensor nodes are tiny and inexpensive devices with limited memory size and interrupted power supply. Thus, they are inappropriate for such heavy-weight computations.

3.4 Network architecture for multichannel communication in WSN

In this section we present different multichannel WSN architectures.

3.4.1 A two-level architecture

A typical WSN architecture is a two-level architecture. There is a powerful entity called the sink which gathers information from sensors. The sink can be equipped with several transceivers. Multiple sinks may be needed in large or heterogeneous networks. In addition, sensors are deployed in the target field to capture and sense the environment. Basically, sensors have a unique radio due to resources constraints. This two-tiers architecture is depicted in Figure 3.3 where sensor nodes are organized in communication islands operating on different channels. Hence, this architecture basically ensures transmission parallelism between the different communication islands. These latter are connected to the sink. They may be structured in subtrees rooted at the sink like in TMCP [Wu 2008] or in clusters like in Zigbee, where each cluster operates independently on a different channel.

Indeed, clustering techniques may be used to achieve energy efficiency, alleviate the traffic generated by the application and thus significantly prolong the network lifetime. In case of channel quality degradation, the whole communication island switches to another available channel.

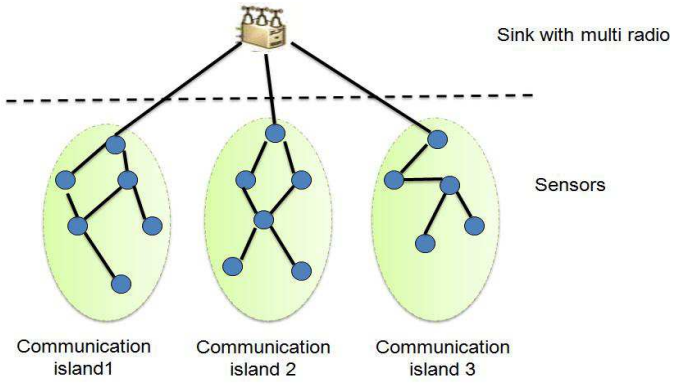


Figure 3.3: A two-level architecture for WSN.

Nevertheless, the two-level architecture lacks the flexibility to support emerging applications that need high throughput. That is why we propose to extend this architecture to a three-level architecture.

3.4.2 A three-level architecture

In a three level architecture, as depicted in Figure 3.4, different types of sensors are distinguished: Some of them manage several radio interfaces, they are called aggregators. These latter are able to listen to several channels simultaneously and communicate with the sink. Other sensors form the lower level of this hierarchical structure. Each of them has one transceiver and can switch between channels associated with their aggregator.

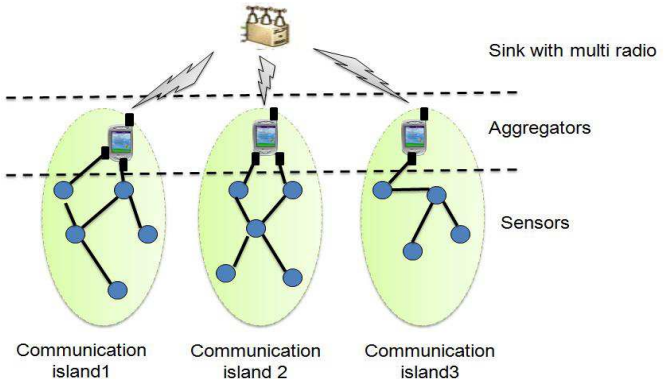


Figure 3.4: A three-level architecture for WSN.

Sensors at the bottom of the hierarchy may form clusters with their associated aggregators to circumvent transmission of huge collected data. This architecture was first introduced in [Li 2008] to maximize network throughput and allow reliable data transmissions in wireless multimedia sensor networks. In [Li 2008], each sensor is associated with one aggregator thanks to a clustering protocol. This latter coordinates the communication in its cluster with a scheduled MAC protocol in order to ensure contention-free intra-cluster communication.

Nevertheless, WSNs need more reliability and are more and more geographically expanded. This situation drive researchers toward a more complex architecture where aggregators form a mesh network and they are not one hop from the sink.

3.4.3 A three-level mesh architecture

As depicted in Figure 3.5, a sensor can change its aggregator if this latter fails or its communication link with its aggregator is broken. This mechanism ensures the auto-adaptivity to failures.

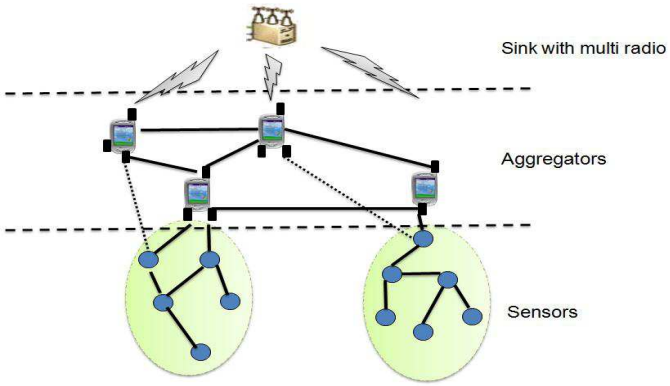


Figure 3.5: A three-level mesh architecture for WSN.

In addition, according to the type of traffic requirements, a sensor chooses the suitable aggregator for flow transmission to meet the desired level of timeliness and ensure bounded delay for real time traffic. A recent work [Zhao 2011] proposes EasiTest, a multi-radio testbed for heterogeneous WSNs. It is suitable for three layer architecture where the first layer is the internet layer, the second layer is a high speed layer composed of EZ271 nodes and finally the third layer includes sensor nodes and relay nodes with limited capabilities. These nodes are called EZ521. They are low speed sensor nodes. Their role is to sense the environment and transmit the sensed data to EZ271 nodes which are considered as sink nodes. Authors claim that EasiTest can be a tool to evaluate novel multi-radios multi-channel protocols.

Actually, the majority of authors use an architecture, suitable for the context of their work, that fits the needed objectives. We favor this three-level mesh architecture, since it is the most general and flexible one.

3.5 Network model for a multichannel WSN

This section highlights the radio propagation, interferences and connectivity models usually considered by the channel assignment approaches in WSNs. Broadly speaking, all the approaches surveyed in this chapter assume the following features of WSNs:
(1) Sensors in the WSN are not mobile.
(2) All sensor nodes are equipped with a single radio with half-duplex radio. Moreover, the sink can be equipped by multiple radio interfaces.

(3) Multiple non-overlapping channels are available.
(4) Only symmetric links are used for communication (e.g., WirelessHART networks [Song 2008]).

3.5.1 Radio propagation

A WSN is generally modeled as an undirected graph $G = (V, E)$ where V is the set of sensors and E is the set of edges. A node, called the sink, is used to process the data sent by the sensor nodes. A communication link $e = (u, v)$ indicates that the packets transmitted by node u may be received by v. The state of the art provides many radio propagation models. In this section, we will detail the unit disk graph and the probabilistic approach.

3.5.1.1 Unit-disk graph (UDG) model

The Unit-Disk Graph, UDG, models the transmission range of a node as an ideal disk of radius 1 centered at the transmitting node in which all nodes inside that disk receive all packets. All sensors outside that disk cannot receive data packets. Consequently, there exists an edge between any two nodes if and only if each belong to the UDG of the other. The UDG model is widely used in the literature [Kim 2008, Tang 2011, Borms 2010]. By definition, the UDG model assumes that any communication link is symmetric.

3.5.1.2 Lognormal shadowing model

The inaccuracy of the unit disk graph model was demonstrated empirically [Baccour 2012]. Indeed, inaccuracy is caused by variance in radio frequency sending power and different path losses depending on the direction of propagation [Zhou 2006a]. For the purpose of modeling reliability, Gallais et al. use the lognormal shadowing model, proposed by Quin and Kunz in [Quin 2003]. In this probabilistic model, communication between a transmitter and receiver has some success probability. This latter depends on the distance between the transmitter and the receiver as depicted in Figure 3.6.

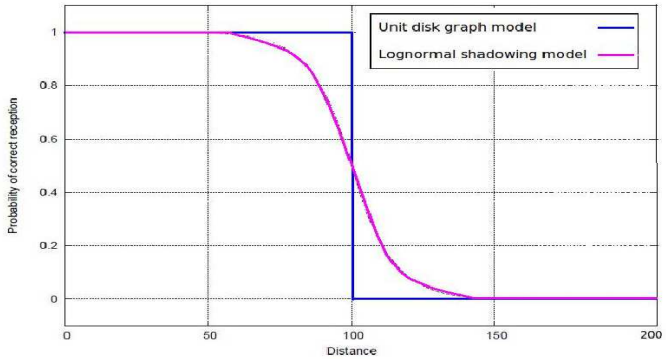


Figure 3.6: UDG and lognormal shadowing models [Ingelrest 2005]

The behavior of this model differs from the unit disk graph model for distances higher than or equal to $R/2$ where R is the transmission range. For such distances, the success

probability decreases progressively from 1 for $R/2$ to 0 for $3R/2$, reaching the value of 0.5 for R.

3.5.2 Interferences

Interferences is one of the key factors that deteriorates the performances and robustness of WSNs. Channel assignment is one solution to alleviate and minimize interferences by using multiple channels. Hence, an interference model defines which set of links can be active (transmission/reception) simultaneously. This section discusses the two interferences models widely adopted in the literature [Al-Ayyoub 2010].

3.5.2.1 Graph-based model

In this model called also protocol model, the interference range of a node is equal to its transmission range. Thus, two edges e_1 and e_2 cannot be scheduled simultaneously if the receiver of at least one link is within the interference range of the transmitter of the other link. We can distinguish two types of interferences under the protocol model as depicted in Figure 3.7:

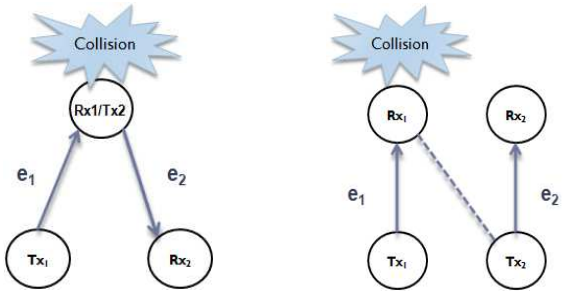


Figure 3.7: Interferences in multichannel WSNs (a) primary interference (b) secondary interference.

1. primary interferences: e_1 and e_2 are adjacent: see Figure 3.7(a)

2. secondary interferences: both receivers of e_1 and e_2 operate on the same channel and at least one of the receivers is within the communication range of the other transmitter: see the dotted link between $Rx1$ and $Tx2$ in Figure 3.7(b) and the collision on $Rx1$.

3.5.2.2 Physical model

In the physical model, the successful transmission over link (i, j) depends on the ratio between the received signal strength at j, the cumulative interferences caused by all other nodes and the ambient noise. Therefore, a packet is received successfully at node j if the $SINR_{ij}$ is greater than a specific threshold.

$$SINR_{ij} = \frac{G_{ij} * P_{ij}}{n_j + \sum_{k,m \in V \setminus \{i,j\}} P_{km} G_{kj}} \tag{3.1}$$

where:

- P_{ij} is the transmission power from node i to node j.
- G_{ij} is the channel gain between node i and node j.
- n_j ambient noise at j.

Although the physical model is more realistic, it is more complicated to implement. Most of the channel assignment techniques assume a graph based model. Some works [Ghosh 2009] use both models.

3.5.3 Connectivity

Many WSN applications require the connectivity between any source node and the sink node. This path may consist of several hops. The connectivity of each node to its next neighbor along the path arbitrates whether or not the data is received successfully at the sink. Moreover, if sensors cannot communicate, directly or indirectly, with the sink, they get isolated and their critical data can never be delivered to the sink. Thus, it is prominent to investigate the connectivity issue in multichannel WSNs.

Compared to single channel WSNs, two kinds of discrepancies arise: (1) a link between two nodes in multichannel WSN disappears if the radios of these two nodes are not assigned to the same channel; (2) several links are present between two nodes if several common channels are assigned to their radio interfaces. In addition, several papers in multichannel WSNs make one of these relevant assumptions when dealing with channel assignment:

- The same topology exists on all channels and this topology remains connected on all channels as depicted in the scheme (1) of the Figure 3.8.
- The topology is connected on all channels whereas the topology may differ from one channel to another as shown in the scheme (2) of the Figure 3.8.
- Multiple channels are needed to have a connected topology as illustrated in the scheme (3) of the Figure 3.8.

A channel assignment technique that tackles interferences and does not take into account the connectivity constraint can lead to disconnected topologies. Consequently, data produced by some sensor nodes cannot reach the sink. The trade-off between avoiding interferences and ensuring connectivity is still a great challenge for researchers.

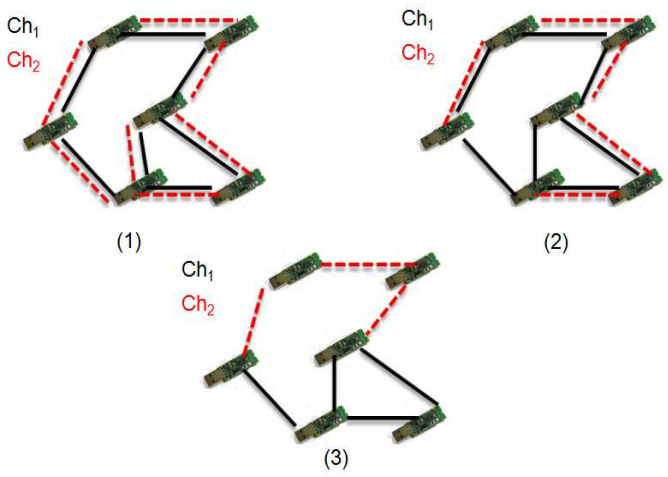


Figure 3.8: Connectivity assumptions in multichannel WSNs.

3.6 Classification of existing multichannel assignment protocols in WSNs

We investigate in this section the existing channel assignment approaches specifically for WSNs, so we restrict our study to single radio nodes. We discuss in detail the channel assignment methods which let a communication take place between two nodes. Finally, we propose our classification of main multichannel protocols found in literature. To this end, we ask three questions:
(1) When or at which frequency is the channel assignment invoked? The answer allows us to distinguish between static, semi-dynamic and dynamic channel assignment methods (see Section 3.6.1).
(2) Which channel is selected? The answer is given by the channel selection policy presented in Section 3.6.2.
(3) How does it work? The answer depends on the channel assignment method itself: see Section 3.6.3.

3.6.1 Categories of channel assignment method

Channel assignment methods can be categorized according to the frequency of channel assignment. We distinguish three categories:

3.6.1.1 Static channel assignment

In many envisioned applications for WSNs, sensor nodes are static on the ground or within structural surfaces. Moreover, WiFi hotspots interference's models are well known. Hence, the environment of the WSN is well known. Furthermore, it is also required to perfectly know the traffic generated by the application. In such conditions, channel assignment can be done once at the beginning of the network deployment, or it can be done very occasionally to adapt to network aging [Galbreath 2006a]. Consequently, this scheme assigns channels to

the radio interfaces, either permanently, or for long durations relative to the channel switching time. This type of assignment is the easiest to implement and no additional switching delay is incurred during data communication. An example is given in [Galbreath 2006b] for wireless in-vehicle communications. The channel assignment protocol is supplied with a map of network fading inside the vehicle. It assigns channels to nodes based on this fading. More generally, static channel assignment is beneficial for delay-sensitive applications, whose environment and traffic pattern are generally known. Static channel assignment has the advantage of a low overhead but can lead to poor performance if the environment in which the WSN operates evolves (e.g. report of an excessive level of interference or jamming). In such a case, a new channel assignment is triggered and will be used for a long period. An example of static assignment method is given by IEEE 802.15.4 [802.15.4], where the network coordinator can decide to switch to another channel if the current one is perturbed: a network reconfiguration is done. Both TMCP [Wu 2008] and MCRT [Wang 2009] adopt the static channel assignment approach.

Although the static scheme works well and has a low overhead, it is not suitable for networks whose links are varying such as WSNs with mobile nodes, or applications whose traffic is unknown.

3.6.1.2 Dynamic channel assignment

Dynamic channel assignment has been proposed as an alternative to static assignment for mitigating the impact of interferences and link dynamics in WSNs. The reasons that motivated researchers to adopt this kind of channel assignment can be classified into two categories: channel variations and channel traffic changes.

- *Channel variations*
  In the industrial applications, reliability is one of the major barriers to the proliferation of wireless communications for sensing and control applications. One major cause of unreliability are interference and radio channel variation [Stabellini 2009]. Indeed, short radio communication sensor nodes are in fact very sensitive to bad channel conditions. Thus, WSNs suffer from link burstiness where packet transmissions from a sender to a receiver are lost during long periods of time. Such spikes of packet losses and radio links intermittency afflict the network causing long delays and instability in communication protocols [Gonga 2012]. In addition, some concerns arise from the harsh nature of wireless channel which let routing topology varies dynamically.

- *Traffic changes*
  The main purpose of multichannel paradigm is to increase throughput. Nevertheless, static channel assignment limits channel utilization. In fact, the traffic passing by a node varies depending on its position in the routing path. For instance, in the data gathering applications, nodes near the sink forward more packets than leaves that are far away.

To improve robustness in multichannel WSNs, multichannel dynamic assignment protocols are introduced to cope with channel variation and traffic changes. However, in the

absence of coordination algorithms, ensuring that two nodes wishing to communicate with each other tune their radio to a common channel results in a low connectivity. To overcome the connectivity problem, most dynamic assignment protocols require tight synchronization of sensors such as Y-MAC [Kim 2008] and MuChMAC [Borms 2010]. Furthermore, high switching delays may still annihilate the use of such strategies in delay critical applications.

In dynamic channel assignment, channel assignment is done very frequently, typically before each transmission. A dynamic assignment allows the assignment protocol to take more accurate decisions (i.e. selection of the best channel based on the knowledge of a more up-to-date information) but with an increased overhead. Dynamic channel assignment has been adopted in several networks deployments [Kim 2008, Borms 2010, Li 2008].

As a conclusion, dynamic channel assignment is suitable for high throughput but non-delay sensitive applications, as well as for mobile WSNs and applications with a varying radio-environment. The new trend in channel assignment protocols favors semi-dynamic channel assignment that represents the best trade-off between low overhead and high adaptivity to channel variety and traffic changes.

3.6.1.3 Semi-dynamic channel assignment

In some scenario, dynamic channel assignment is not appealing because frequent interface switchings introduce additional delays that impact the freshness of the data delivered.

The semi-dynamic channel assignment is done periodically or event-based. We distinguish two trends:

1. *Approaches assuming an homogeneous network:* such family includes many protocols such as MMSN [Zhou 2006b], TACA [Wu 2009], EM-MAC [Tang 2011], RMCA [Yu 2010b] and ARM [Li 2010]. This list is not exhaustive but we highlight the most relevant ones.

2. *Approaches making distinction between sensor nodes:* Sensors having the heaviest traffic loads follow static channel assignment whereas other sensors pursue dynamic channel assignment. Consequently, in this scheme, a subset of the available channels and radios are assigned statically and the rest of the spectrum is available for dynamic assignment to the other radios [Kyasanur 2006]. Another example is given by [Fotue 2012] where a set of parents and leaves are assigned to a single fixed channel while specific sensors, called mediators, are assigned to several orthogonal channels. Hence, they switch to the static channels assigned to their parents to collect data.

The semi-dynamic approach allows more adaptivity to traffic changes and interferences while coping with long switching delays. This approach is convenient for application that requires a high throughput and also for delay-sensitive applications like for instance interactive gaming.

3.6.2 Channel selection policy

Whatever the channel selection policy, and as long as the number of available channels is not exhausted, a node should never be granted the channel already allocated to one of its conflicting nodes. Two nodes are said conflicting if and only if they cannot transmit simultaneously on the same channel without perturbing each other. Usually, protocols consider that nodes up to 2-hop are conflicting. The channel is generally selected from a Preferable Channel List [Crichigno 2008], denoted PCL. To avoid the use of a busy channel at the receiver, this list is computed either by the receiver like in EM-MAC [Tang 2011] or by exchanging the sender and receiver lists like in MMAC [So 2004a]. Two issues arise: Which channel is (1) inserted in PCL? (2) which channel is selected from PCL?

For the first issue, we distinguish solutions that are:

- *counter-based*: a counter is maintained per channel taking into account its recent history. If the value of this counter reaches a low threshold, the corresponding channel is removed from PCL. For instance, in EM-MAC [Tang 2011], initially all channels have the maximum quality value. This value is decreased each time a bad event occurs (e.g. non receipt of an ACK after the maximum number of transmissions). If the quality value reaches a low threshold, the corresponding channel is inserted in the blacklist for a given duration. The PCL contains all the available channels that are not in the blacklist.

- *channel quality indicator-based*: the selection of the best quality channel rises the problem of channel quality indicators. Many indicators have been used, such as LQI, packet loss, available throughput, the received signal power and the signal-to-noise ratio (SNR), etc. They are based on periodic measures to estimate the channel quality. Only channels whose quality is over a given threshold are inserted in the PCL.

- *hybrid*: a counter is maintained based on different measures of the channel link quality.

For the second issue, the selection of a channel from the PCL, different policies are possible:

- *round robin* like in EM-MAC [Tang 2011]: a node switches between channels based on its pseudorandom channel schedule.

- *least chosen channel*: if all available channels have been picked up by at least one one-hop or two-hop neighbor, the node randomly selects one of the least chosen channels. Then, this choice is broadcast to its neighbors up to 2-hop.

- *least loaded channel*: the aim of this policy is to minimize the maximum load of each available channel from the PCL within two-hop neighbors. It has been proved that high communication latency and more radio collisions occur in loaded channels [Wu 2009].

- *probabilistic*: protocols rely on a probability based random channel selection to avoid the hidden nodes specifically when traffic load is heavy. RMCA [Yu 2010b] and ARM [Li 2010] are examples of protocols that obey to this scheme.

Dynamic channel selection is crucial to avoid channels that are congested or degraded ones by interferences or jamming attacks. In addition, it achieves load balancing among available channels providing better adaptation to traffic pattern changes during network lifetime.

3.6.3 Channel assignment methods

Communication between nodes allowed to switch from one channel to another is possible if and only if both the sender and the receiver are on the same channel during the transmission time. Channel assignment methods differ in how devices negotiate the channel to use and solve medium contention. The channel assignment can be:

- **Implicit** when nodes implicitly agree on 1) the channel to switch and 2) when the channel switching occurs. Among protocols using implicit assignment, we distinguish those:

  - *Based on islands of communication*: Historically, the first multichannel assignment protocols are based on islands of communication. The sink plays the role of gateway between these islands. Each island of communication uses its own channel for internal communications. This channel is different from the channels used by neighboring islands of communication. This is how a PAN coordinator may manage several islands of communication each on its own channel in IEEE 802.15.4. Another example is given by TACA [Wu 2009], where the sink assigns to each child a distinct channel. This channel is used for any communication within the subtree rooted at this child. In TMCP [Wu 2008], the sink divides the network into multiple subtrees, allocates different channels to each subtree in a receiver-centric way and forwards data flows only along its corresponding subtree. The key idea behind these protocols is to minimize data gathering delays by avoiding time expensive channel switchings along the path. Spatial reuse of the bandwidth within an island of communication should be provided by the MAC protocol, like for instance in OCARI [Mahfoudh 2010a] with node coloring.

  - *Using frequency hopping*: Nodes hop from channel to channel according to:

    - either a common hopping sequence generally given by a centralized entity (e.g. the master of the considered Piconet in Bluetooth [Bluetooth]). Although some protocols were proposed for ad hoc networks like CHAT [Garcia-Luna-Aceves 2000], common hopping is not frequently used in WSNs.
    - or independent hopping sequences. Each node has its own hopping sequence. However, these sequences are all built according to the same pseudo-random model and differ only by the seed that depends on the node address, like in EM-MAC [Tang 2011].

    The common hopping sequence is simpler but more vulnerable. The independent hopping sequences require more computation and storage at each sender node if the channel assignment is receiver-based like in EM-MAC [Tang 2011]. The main advantage of frequency hopping is its better immunity to noise. In [Gonga 2012], authors argue that for single-hop networks, channel hopping (with more than two

channels) decreases packet loss correlation. Its drawback is the need for a tight node synchronization.

- **Explicit** when nodes decide to negotiate channel selection or use scheduling schemes to coordinate channel switching. We distinguish different techniques:

  - *Dedicated control channel*: one channel is intended exclusively for control traffic, whereas the remaining channels are used for data traffic. Each node listens to the control channel to know the channel it must switch to for its communication to take place. An example is given by Y-MAC [Kim 2008] and ARM [Li 2010]. This method is easy to implement and does not require tight time synchronization between nodes. However, this mechanism can drastically reduce bandwidth use efficiency if the amount of data exchanged after each rendezvous or the number of available channels is small.

  - *Splitting phase*: time is split into alternating periods of control and data phases. In a control phase, all nodes listen to the control channel to make an agreement. In the data phase, sensors switch to their respective channels negotiated in the previous control phase to exchange data in parallel. Notice that unlike the dedicated control channel protocols, the splitting phase protocols may use the control channel also for data transmission. An example is given by MMSN [Zhou 2006b] for a contention based MAC protocol and by Y-MAC [Kim 2008] for a scheduled access MAC protocol. The crucial asset of this technique is to solve the deafness and hidden node problems. However, it may suffer from channel inefficiency: many channels remain unused in control phase.

  - *Joint channel and time slot assignment*: New approaches solve jointly channel and time slot (ie. opportunity to access the medium without collisions) assignment to achieve reliable and real-time communications and to fully utilize the expanded bandwidth. In WirelessHART [Song 2008] and Y-MAC [Kim 2008] communications are scheduled based on Time Division Multiple Access (TDMA) to arbitrate communications between devices. To enhance reliability, WirlessHART and Y-MAC employ a channel-hopping scheme.

    The MuChMAC protocol [Borms 2010] combines TDMA and asynchronous MAC techniques. It is a receiver-based channel assignment: A node is able to independently choose its receiving channel switching sequences based on its identifier and the current slot number using a pseudo-random generator.

    The above mentioned solutions are based on heuristics or approximations. Few of them afford optimal solutions in particular scenarii. For instance, Incel et al. [Incel 2012], propose an optimal convergecast scheduling algorithm JFTSS in multichannel WSNs, on any network topology where the routing tree has an equal number of nodes on each branch.

  - *Joint channel assignment and routing*: Several proposals are modifying the MAC layer to find the best channel for a flow in collaboration with the network layer. Since,

frequent channel switchings along the path taken by a message increase the end-to-end delay, channel assignment and routing are tackled jointly in order to decrease end-to-end delays. More precisely, both the channel assignment plan process and routing protocol are invoked in case of network topologies changes. This is justified because the routing protocol needs to select not only the optimal path in-between different nodes, but also the most appropriate channels on the path.

[Yen 2009] is the first work that designs an integrated channel assignment and convergecast routing algorithm. They propose an iterative algorithm called ICADAR that can find another routing path if the channel assignment method fails to identify a feasible channel assignment on the original path. Recently, Li et all. [Li 2011a] propose a routing scheme in multipower multiradio WSNs which tackles jointly the scheduling, channel assignment and power control problem. Therefore, they design an efficient heuristic algorithm based on random walk to obtain efficient paths while minimizing the energy consumption and the end-to-end delay along paths. Dynamic channel allocation is used for each link.

- *Game theory*: As WSNs stretch to be more and more decentralized and auto-adaptive, nodes can be seen as autonomous agents. In order to solve the challenging multichannel assignment problem, all sensor nodes are modeled as players and the available non-overlapping channels used by nodes to receive packets are the actions of sensor nodes. The target of the game can be for example the reduction of the total interference in the network. Each player picks up a channel different from its interfering players. The panoplies of player strategy constitute the channel coordination and assignment [Yu 2010a], [Yu 2010b]. This innovative technique has the advantage of being highly distributed and requiring the exchange of limited information to judiciously assign channels such as RMCA [Yu 2010b]. Nevertheless, convergence of game based protocols is still nontrivial.

- *Coloring based*: A channel assignment problem is typically modeled as a graph coloring problem. Vertices, edges and colors represent respectively sensor nodes, potential interferences and the number of non overlapping channels. The purpose is to maximize parallel transmissions and minimize interferences. The technique consists in covering all sensor nodes (vertices) with the minimum number of channels (colors) such that adjacent nodes have different channels. This approach was addressed in MMSN [Zhou 2006b] and TACA [Wu 2009]. This technique entails the pros and cons of node coloring schemes. A drawback of this technique is the high number of channel switches needed by an upstream message from a sensor node to the sink as well as by a downstream message.

3.6.4 Discussion

In this subsection, we first study the relationships between channel assignment categories and channel assignment methods.

- *frequency hopping* is used in dynamic channel assignments, although it could be used also in semi-dynamic assignments depending on the duration between two successive frequency hops (i.e a single transmission or the awake cycle). The experimental studies [Gonga 2012, Ortiz 2010] assess the utility of multichannel paradigm for reliable communication. Authors focus on channel hopping as most multichannel MAC protocols rely on it for channel diversity. In [Ortiz 2010], authors argue that in single channel WSNs, route diversity bids the same level of reliability as channel diversity. This assessment is highlighted also in [Gonga 2012]. Authors establish that for one-hop communication or fixed topologies, channel hopping provides a substantial improvement on reliability, higher than adaptive routing. However, in dense and medium dense deployments, adaptive routing and channel agility provide the same level of end-to-end reliability.

- *dedicated control channel* can be used in the three types of channel assignments.

- *splitting phase* can be used in semi-dynamic and dynamic assignments.

- *game theory and node coloring* can be used in semi-dynamic assignment. Their use in dynamic environment does not seem realist as long as the overhead induced is too high.

We now discuss the relationship between channel assignment categories and WSN architecture.

- *For the two-level architecture,* we can distinguish two cases:

  1. One channel per communication island. The channel assignment to a communication island can be static like in TMCP [Wu 2008] or semi-dynamic to adapt to changing channel quality. A dynamic channel assignment is also possible, provided that the sets of frequencies used by interfering communication islands at any time are different. Parallel transmissions are also possible between:
     - Communication islands with different channels
     - Non interfering communication islands operating on the same channel.
     - Non interfering nodes within the same island.

     The basic advantage of this structure is the absence of switching overhead. The shortcomings come from the fact that there is only a single channel to support the traffic load generated in an island. Hence, this architecture is not adapted to applications with high throughput requirements.
  2. Several channels are allocated to a communication island. Like previously, the channel assignment may be static, semi dynamic. It may be also dynamic like in EM-MAC [Tang 2011]. This architecture supports higher application throughput due to the simultaneous use of several channels within the same communication island. The channel assignment is more complex than in the first case. Moreover, the end-to-end delays are increased due to the switching delays encountered along the path to the sink.

- *In the three-level architecture,* any aggregator should operate on different channels simultaneously, otherwise this architecture does not offer any benefit with regard to a two-level architecture. This simultaneous use of several radios at the aggregator level will reduce the traffic bottleneck. The three channel assignment categories can be used with the three-level architecture.

- *In the three-level mesh architecture,* a static channel assignment is not suitable according to its functionality described in Section 3.4.3,

3.7 Cross-layer design for multichannel allocation and routing

Most common proposed assignment schemes were not designed to differentiate among traffic patterns. During network lifetime, sensors are transmitting a wide variety of flows, each of them with very different QoS requirements. Many studies have tried to provide channel allocation strategies with methodologies to differentiate QoS. For instance, JSAR [Slimane 2011] satisfies multiple QoS requirements.

According to flow priority, a specific metric is optimized: for high priorities (hard real time constraint) the end-to-end delay is taken into account. For lower priorities, average energy consumption and residual energy per path are considered.

We think that cross layering paradigm with the upper and lower layers could be helpful to meet the QoS required by the application in resource constrained networks. According to the application requirements, the network layer takes its decision based on the environment information provided by its MAC layer. Furthermore, we propose that delay sensitive flows are assigned to protected channels that present a low interference level and minimal traffic load. Whereas, non-constrained traffic flows are assigned to channels in such a way that probabilistic QoS guarantees are ensured.

3.8 Taxonomy proposed

The taxonomy of multichannel protocols we propose is based on the four questions: 1) what is the goal? 2) when channel assignment is done? 3) which channel is selected? and 4) how channel assignment is done? Now, we can analyze existing multichannel assignment protocols in WSNs.

Table 3.1: Classification and analysis of existing multichannel assignment protocols.

		Static		Semi-dynamic				Dynamic			
		TMCP [Wu 2008]	MCRT [Wang 2009]	MMSN [Zhou 2006b]	TACA [Wu 2009]	EM-MAC [Tang 2011]	RMCA [Yu 2010b]	[Li 2011a]	Y-MAC [Kim 2008]	MuChMAC [Borms 2010]	COM-MAC [Li 2008]
Goals	Parallel transmissions	√		√		√		√			√
	Increase capacity				√			√			√
	Enhance robustness				√	√	√				
	Reduce delays	√	√					√			
Properties	Autoadaptive				√						
	Broadcast	√		√					√	√	
	QoS		√		√		√				
	Energy efficient							√	√	√	√
	Channel switching	√	√								
	Reduced interf.	√	√		√	√					
	Scalable							√	√		√
Channel Selection	RR					√			√	√	
	least chosen			√							
	least loaded				√						√
	probabilistic						√	√			
Channel assignment method	Dedicated ctl channel								√		
	Splitting phase										√
	Game theory						√				
	Node coloring			√	√			√			
	Communication island	√	√								√
	Frequency hopping					√			√	√	
	Joint CA & scheduling										
MAC		CSMA/CA	CSMA/CA	CSMA/CA	CSMA/CA				TDMA	TDMA XMAC	
Layer in charge of channel allocation		MAC	Routing	MAC	Routing	MAC	MAC	Routing	MAC	MAC	MAC
Solution maturity		simul testbed	simul	simul	simul	testbed	simul	simul	simul	testbed	simul
WSN architecture		2-level	2-level	2-level	2-level	2-level	2-level	2-level	2-level	2-level	3-level
Centralized / distributed solutions		centr.	centr.	distrib.	distrib.	distrib	distrib.	distrib.	ditrib.	distrib.	distrib.

As depicted Table 3.1, channel assignment methods are classified into three categories: static, semi-dynamic and dynamic. Static approaches are intuitively blind and cannot accommodate changes. Accordingly, they can be used in scenarii where the environment and the application (i.e traffic pattern) are well-known and do not vary along time. For the dynamic approach, the typical scenario is when the radio environment and application traffic are unpredictable and change along time.

The semi-dynamic approach can be considered when the environment and the traffic remain stable for a period of time (e.g we can define application phases, during which the traffic pattern and the environment are known). Mo Sha et all. [Sha 2011b] have addressed the application of WSNs in Home Area Networks (HAN). They proposed ARCH [Sha 2011a] a distributed and adaptive channel hopping for HAN applications that can handle effectively dynamic channel conditions in apartments.

Centralized solutions, JFTSS [Incel 2012], provide optimal or near optimal channel assignment, because the entire network information is available. However, they are not scalable. On the other side, distributed approaches, like EM-MAC [Tang 2011] and MMSN [Zhou 2006b], adapt themselves more quickly to network changes, node failures and variation of traffic profiles.

3.9 Conclusion

In this chapter, we have considered the challenges raised by multichannel communications, mainly frequency of channel assignment, channel selection policy and channel assignment method. We have also proposed WSNs architectures for multichannel communications. Then, we have given a comprehensive classification of well known multichannel assignment protocols in WSNs. This taxonomy is illustrated in a recapitulative table highlighting the main features of some surveyed protocols.

As argued, resorting to multichannel communication ensures higher network capacity and robustness against interferences.Thus, in Part II of this thesis report, we will investigate the problem of the minimizing the time delivery of data in bandwidth-limited WSNs by combining multichannel paradigm and interference-aware TDMA scheduling.

Part II

Joint Multipath Routing and Scheduling in Multichannel WSNs

Chapter 4

Lower Bounds for Convergecast: Theoretical Study

Contents

4.1 Introduction and motivations 45
4.2 Network models and assumptions 47
4.2.1 System model 47
4.2.2 Definitions 48
4.2.3 Assumptions 48
4.2.4 Modeling interferences for data gathering 49
4.2.5 Notations 51
4.3 Theoretical bounds for homogeneous traffic 51
4.3.1 Linear networks 52
4.3.2 Tree networks 52
4.3.3 Optimal schedule 52
4.4 Theoretical bounds for heterogeneous traffic 56
4.4.1 Without immediate acknowledgment 56
4.4.2 With immediate acknowledgment 59
4.5 Conclusion 63

4.1 Introduction and motivations

Data gathering applications represent the typical applications supported by WSNs. Data from sensors are collected over a tree rooted at a special entity, called the sink, which is generally more powerful than the other nodes. This many-to-one communication paradigm is called **convergecast**. When the volume of data transmitted by any sensor is reduced, aggregation techniques are used to increase network efficiency and throughput. When several

samples are transmitted in a single MAC (Medium Access Control) frame, the length of the frame is usually close to the maximum length allowed by the MAC protocol (e.g., 127 bytes for the IEEE 802.15.4 MAC protocol). As a consequence, no aggregation is possible in the intermediate nodes (*i.e.*, **raw data convergecast**). We observe that nodes close to the sink have a higher traffic demand. Hence granting an equal number of slots to each node is not adequate. That is why we investigate the problem of convergecast scheduling that ensures to any node a channel access that is proportional to its traffic demand. In addition, these applications generally require small delays for data gathering and time consistency of the data gathered. This time consistency is usually achieved by a small gathering period. As argued in the previous chapter, using multichannel techniques ensures parallel transmissions and higher capacity. Therefore, we focus on raw data convergecast in multichannel WSNs to ensure smaller gathering delays and a higher throughput.

Under heavy traffic conditions which drastically increase the probability of collisions and retransmissions, contention-based medium access protocols are inefficient for periodic data collection. In contrast, Time Division Multiple Access, TDMA, is a contention-free protocol where time is divided into cycles. A cycle is divided into slots. Interfering nodes are scheduled to transmit in different slots. Each node transmits data in its allocated slots. TDMA is the preferred access scheme for applications that require energy efficiency and bounded end-to-end delays. On the one hand, since the TDMA protocol removes idle listening and overhearing, which are the main sources of energy drain, TDMA deterministic scheduling is appropriate for low power devices since nodes turn off their radio in non scheduled time slots ensuring energy efficiency and prolonging network lifetime. On the second hand, TDMA protocols have the ability to deliver packets with deterministic delay bounds by eliminating collisions and retransmissions. Indeed, WirelessHART [Song 2008], a standard for control applications, uses a TDMA data link layer to control medium access.

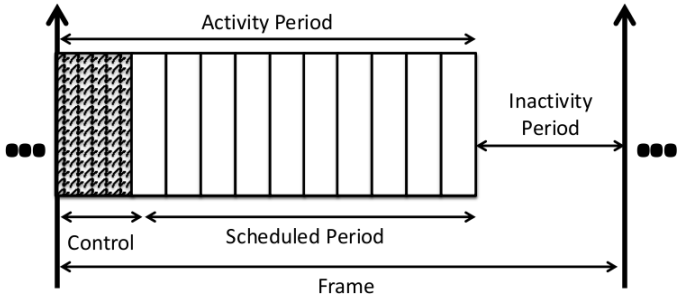


Figure 4.1: A data gathering frame.

Figure 4.1 depicts a data gathering frame. The frame can repeat over time, defining a cycle. The cycle length is equal to the duration between the beginnings of two successive frames, and it should meet the application requirements concerning data gathering delays. The frame consists of an activity period and an inactivity period. The activity period is subdivided into a control period, where nodes exchange control messages (e.g., hello messages to discover the neighborhood), and a scheduled period, where nodes transmit the user data towards the sink, according to pre-established scheduling. In the inactivity period, all nodes sleep to save energy, and no transfer is therefore possible in this period. Our goal is to

ensure that the data gathering is achieved in a single cycle and that all the packets generated in a cycle can be transmitted in a single cycle to the sink, assuming that all packets are periodically generated. With FIFO (First In First Out) queues, the worst case delay for delivering to the sink data generated by a sensor is equal to cycle-length + activity-period. This delay is obtained when the sensor whose transmission slot is the first in the activity period misses its opportunity to transmit and must wait until the next cycle to transmit its generated data. Moreover, in the worst case, this packet is received by the sink in the last slot of the activity period. Notice that for any intermediate node, the message is always received in a slot preceding the slot where this node is scheduled to transmit. Hence, no additional delay is introduced by the forwarding.
Furthermore, we notice that in a single channel context, the transmissions of interfering nodes are scheduled in different slots; whereas in a multichannel context, the transmissions of interfering nodes are scheduled in the same slot, but on different channels, provided that both the sender and the receiver have an available radio interface. As a result, the throughput is increased. In addition, the throughput can be further improved by equipping the sink with multiple radio interfaces.

Therefore, we tackle in this chapter the problem of computing the minimal time to complete convergecast ensuring to any node a medium access that is proportional to its traffic demand. Unlike most previous studies, we consider both cases: homogeneous and heterogeneous traffic. Furthermore, two cases of transmissions are distinguished: without acknowledgment and with immediate acknowledgment. We provide the minimum bound on the number of time slots needed for a convergecast in a multichannel environment with a sink equipped with multiple radio interfaces. Indeed, computing the minimum number of needed slot helps us to evaluate how close designed algorithms can get to the known theoretical lower bounds.

4.2 Network models and assumptions

In this section we will state our assumptions and models that we adopt to determine the lower bounds on the schedule length for raw convergecast.

4.2.1 System model

We model a WSN netwok as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ denotes the set of sensors and $\mathcal{E}$ represents the set of edges. We adopt the Unit-Disk graph model (see Chapter 3): Under a UDG model, there exists an edge between every two nodes that are at most a unit distance apart from each other. By definition, the UDG model assumes that any communication link is symmetric. A communication link $e = (u, v)$ indicates that the packets transmitted by node u may be received by v. There is only one sink. Both sensors and sink remain static after the deployment. All nodes except the sink are transmitters. Moreover, we assume the simple interference model: the graph-based (protocol) model (see Chapter 3). We suppose that the interference range of a node is equal to its transmission range.

We also assume that the routing tree is given. This routing tree is rooted at the sink and allows each node to reach the sink using the path of minimum cost (e.g the shortest path).

4.2.2 Definitions

We first introduce some definitions:

Definition 1. *We focus on a raw data convergecast in a WSN where* ***nchannel*** > 1 *channels are available at each node.*

Definition 2. *We define* ***ninterf*** *as the number of radio interfaces of the sink,* $ninterf \geq 1$*. Any other node has only one half-duplex radio transceiver which means that a sensor cannot transmit and receive in the same slot.*

Definition 3. *For any node* u*, we define* ***Parent(u)*** *the parent of* u *in the routing tree. Similarly,* ***Children(u)*** *denotes the set of children of* u *in the routing tree. We denote* ***Subtree(u)*** *the subtree of the routing tree that is rooted at node* u*.*

Definition 4. *For any node* u*, we define* ***Gen(u)*** *the number of packets generated by* u *to transmit its own data. This number of packets may differ from one node to another, that is why we speak of heterogeneous traffic. Similarly,* ***Trans(u)*** *denotes the total number of packets transmitted by* u *in a data gathering frame. This number includes the packets generated locally by* u *to transmit its own data and the packets received from its children.*

From definitions 3 and 4, we obtain:

$$Trans(u) = \sum_{v \in Subtree(u)} Gen(v)$$

.

Definition 5. *For any node* u*, we define* ***Conflict(u)*** *the set of nodes* v *that are not allowed to transmit in the same slot and on the same channel as* u*. Notice that in a raw data convergecast, all the data transmissions are from a node to its parent in the routing tree. Let* ***nchild*** *denote the number of children of the sink.*

Definition 6. *Acknowledgment policy: we consider two policies:*

- *either there is no acknowledgment,*
- *or the acknowledgment is sent in the same time slot and on the same channel as the packet it acknowledges.*

4.2.3 Assumptions

We adopt the following assumptions:

Assumption 1. *The network topology and the routing tree associated with the raw data gathering are given.*

Assumption 2. *For any node* $u \neq sink$*,* $Gen(u) > 0$ *and these* $Gen(u)$ *packets are present in the buffer of* u *at the beginning of the raw data gathering.*

Assumption 3. *For the sake of simplicity, the size of time slots is constant and enables the transmission of one packet. If the immediate acknowledgment policy is used, the size of the slot enables the transmission of one packet and its acknowledgement.*

Assumption 4. *A node that receives a paquet in a slot t, may forward it in the next slot $t+1$ if necessary.*

We also adopt some additional assumptions that are used only for the computation of theoretical bounds and for the performance evaluation in next chapters.

Assumption 5. *No message loss and no node failure.*

Assumption 6. *No topology link in addition to those belonging to the routing tree.*

4.2.4 Modeling interferences for data gathering

For any node u, the determination of the set $Conflict(u)$ depends on the policy used for the acknowledgment (immediate acknowledgment or no acknowledgment).

Lemma 1. *In a data convergecast and in the absence of acknowledgment, $Conflict(u)$ contains:*

a) the node u itself,

b) the node $Parent(u)$,

c) all the children of u,

d) all the nodes that are 1-hop away from $Parent(u)$,

e) all the nodes whose parent is 1-hop away from u.

Proof. Any node $u \neq sink$ cannot on the same channel simultaneously:

- perform two point-to-point transmissions to two different nodes: case 4 in Figure 4.2.
- transmit and receive: case 3 in Figure 4.2.
- receive from two different children: case 6 in Figure 4.2.

These rules apply also for $Parent(u)$ (see case 5 for simultaneous transmission and receipt and case 1 for simultaneous receipts in Figure 4.2). In addition, when u transmits, it interferes with the transmission of any node v, whose parent is one-hop away from u (see case 2 in Figure 4.2). □

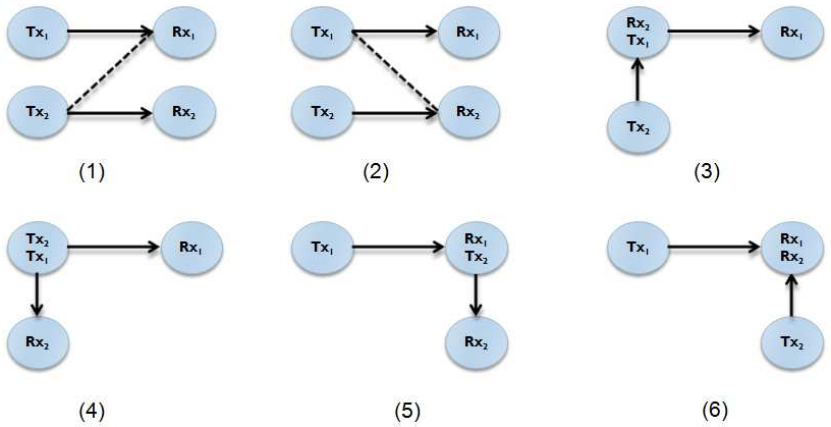


Figure 4.2: Conflicting transmissions without acknowledgment.

Lemma 2. *In a data convergecast with immediate acknowledgment, $Conflict(u)$ contains:*

a) the node u itself,

b) the node $Parent(u)$,

c') all the nodes that are 1-hop away from u or $Parent(u)$,

d') all the nodes whose parent is 1-hop away from u or $Parent(u)$.

Proof. The set $Conflict(u)$ with immediate acknowledgment is equal to the set $Conflict(u)$ without acknowledgment $\bigcup$ the set of nodes that are 1-hop away from u (see case 8 in Figure 4.6) $\bigcup$ the set of nodes whose parent is 1-hop away from $Parent(u)$ (see case 7 in Figure 4.6). □

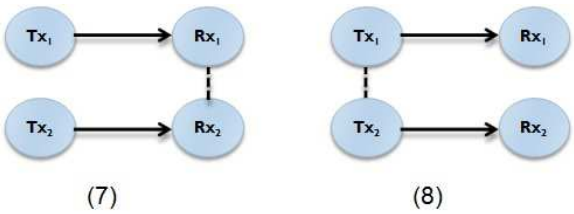


Figure 4.3: Additional conflicting transmissions with immediate acknowledgment.

Corollary 1. *With the immediate acknowledgment policy, $Conflict(u)$ contains the two additional cases depicted in Figure 4.6.*

Since we have shown in Corollary 1 that for any node u, the set $Conflict(u)$ with immediate acknowledgment is a superset of $Conflict(u)$ without acknowledgment, the number of slots for a raw data convergecast is higher than or equal to this obtained with the immediate acknowledgment policy. Hence, the lower bounds computed without acknowledgment still hold for immediate acknowledgment, but possibly with a lesser accuracy.

We give these bounds for homogeneous traffic in Section 4.3 and for heterogeneous traffic in Section 4.4. To do that, we introduce some additional notations.

4.2.5 Notations

The notations introduced here will be used throughout this chapter as well as the rest of the manuscript.

We order the children of the sink by the decreasing order of the number of their transmissions. The first child transmits more messages to the sink than any other. It is called the most transmitting child. We denote:

- $g = min(nchild, nchannel, ninterf)$,
- $S_n = \lceil \frac{\sum_{u \in WSN} Gen(u)}{g} \rceil$. We designate by $S_n^1 = \lceil \frac{N-1}{g} \rceil$, the value of S_n when each node has one packet to transmit where N is the number of nodes including the sink.
- $S_t = Gen(ch1) + 2 \sum_{v \in\ Subtree(ch1), v \neq ch1} (Gen(v)) + \delta$, where $ch1$ is the most transmitting child of the sink and $\delta = 1$ if the $(g+1)^{th}$ child of the sink requires the same number of transmissions as the first one, and $\delta = 0$ otherwise.

  We denote by $S_t^1 = 2n_1 - 1 + \delta$, the value of S_t when each node has one packet to transmit. n_1 is the number of nodes in the subtree rooted at $ch1$ and $\delta = 1$ if the number of nodes of the $(1+g)^{th}$ most populated subtree rooted at a sink child is equal to the number of the nodes of the subtree rooted at $ch1$, 0 otherwise.

4.3 Theoretical bounds for homogeneous traffic

The following fundamental question: "*what is the minimum number of slots we need to collect raw data from a WSN organized in a tree?* " has been investigated in many studies. Nevertheless, they have specifically target the simple case when sensors generate only one packet. In [Zhang 2009], authors address jointly the link scheduling and channel assignment for convergecast in networks operating according to the WirelessHART standard. Authors have proved that for linear networks with N single buffer devices, the minimum schedule length obtained is $(2N-1)$ time slots with $\lceil N/2 \rceil$ channels.

Incel et all [Incel 2012], have proved that if all interfering links are removed (with the required number of channels), the schedule length for raw-data convergecast is lower bounded by $max(2n_k - 1, N)$ where n_k is the maximum number of nodes in any top-subtree of the routing tree and N is the number of source nodes.

Our results given in this section **are extension of already published results, because we consider the case where the sink is equipped with multiple radio interfaces** and at least two channels are available at each node.

Theorem 1. *In any WSN with N nodes, a lower bound on the number of slots required by a raw data convergecast is S_n^1.*

Proof. In any network with N nodes, the sink has to receive $N-1$ messages from its children. The number of simultaneous transmissions to the sink is limited by the number of children and the number of sink interfaces as well as the number of available channels (each

interface using its own channel). Hence, the number of slots needed is higher than or equal to $\lceil \frac{N-1}{min(nchild,nchannel,ninterf)} \rceil$. Hence the theorem. □

4.3.1 Linear networks

Theorem 2. *In linear networks where each node has nchannel* > 1*, the minimum number of slots for a raw data convergecast is* $2N-3$*, where* N *is the number of nodes including the sink, whatever the number of interfaces of the sink.*

Proof. Consider a linear network with N nodes, where nodes are numbered from 1 to N, starting by the sink node. Any node $i > 1$ is at a distance $i - 1 < N$ from the sink. Node 2 needs to transmit $N - 1$ packets to the sink (node 1) and needs to receive $N - 2$ packets from node 3. Since node 2 has a single interface, these transmissions cannot overlap. As a consequence, a lower bound on the number of slots is equal to $N-1+N-2 = 2N-3$.

This bound is reached by the algorithm that schedules:

- in any odd slot, any node that is $(2h+1)$-hop away from the sink, with $h \in [0, \lfloor \frac{N-1}{2} \rfloor]$;
- in any even slot, any node that is $2h$-hop away from the sink, with $h \in [1, \lfloor \frac{N-1}{2} \rfloor]$.

Hence the theorem. □

4.3.2 Tree networks

Theorem 3. *In tree networks, a lower bound on the number of slots for a raw data convergecast is Max(*S_n^1*,*S_t^1*).*

Proof. The sink requires $\lceil \frac{N-1}{g} \rceil$ time slots to receive all the packets generated in the network as seen in Theorem 1. Furthermore, each child of the sink has at least one packet to transmit. Moreover, let us consider, $ch1$, the child of the sink with the highest number of descendants. Let n_1 be the number of nodes in the subtree rooted at this child. This latter requires (n_1-1) slots to receive the packets from its children and n_1 slots to transmit its packets to the sink. Since all these transmissions are sequential, at least $(2n_1-1)$ slots are needed. If the $(g+1)^{th}$ most populated subtree has a number of nodes equal to n_1, then its schedule will require an additional slot. Indeed the schedule of this subtree requires the same number of slots as the first one. However, the $(g+1)^{th}$ sink child starts to transmit at the second slot, that is a slot later. Indeed, at the first slot, all the available interfaces of the sink are used by the g children of the sink having the most populated subtrees. Consequently, the schedule of this subtree will end a slot after. Hence, the value of δ and the theorem. □

Notice that a multi-line topology can be seen as a specific case of a tree topology.

4.3.3 Optimal schedule

In this section, we build an algorithm that reaches the bounds given in the previous theorems. As a consequence, these bounds are optimal as well as the algorithm. The basic idea of this

optimal algorithm is to maintain the $g = min(nchild, nchannel, ninterf)$ interfaces of the sink busy as long as possible. We first notice that for a linear topology, the algorithm given for the proof of Theorem 2 meets this requirement. We now extend this algorithm to tree topologies. This algorithm, called FlipFlop, proceeds as follows:
First, it orders all the subtrees rooted at a sink child according to the decreasing number of nodes. The g first subtrees form the first group, the next g subtrees form the second group and so on until the last one. We first consider the case where $g \leq nchild \leq 2g$, there are at most 2 groups.
This algorithm schedules in the odd slots:

- the nodes of odd depth in the subtrees 1 to g, including the sink children belonging to the first group,
- the nodes of even depth in any other subtree.

It schedules in the even slots:

- the nodes of even depth in the subtrees 1 to g,
- the nodes of depth 1 in subtrees $g+1$ to $2g$,
- the nodes of odd depth > 1 in any subtree $> g$.

Furthermore, this schedule meets the following rules:

- If several nodes have the same parent that is not the sink, they are scheduled round robin.
- In the same subtree, two 2-hop nodes that are scheduled in the same time slot transmit in different channels. For instance, channel 1 is used at depth 1, 5, and 9, whereas channel 2 is used at depth 3, 7 and 11...
- As soon as the schedule of a subtree is completed, the first sink child that has never been scheduled is scheduled.

A slot where the g interfaces of the sink are not busy is said incomplete.

Theorem 4. *In a multichannel WSN the optimal number of slots for a raw data convergecast is:*

- $2n_1 - 1$ *if* $nchild = g$;
- $2n_1 - 1 + \delta$ *if* $g + 1 \leq nchild \leq 2g$;

with $g = min(nchild, nchannel, ninterf)$, n_1 *is the number of nodes in the subtree rooted at* $ch1$, $nchild$ *is the number of sink children and* $\delta = 1$ *if the number of nodes of the* $(g+1)^{th}$ *most populated subtree rooted at a sink child is equal to* n_1, 0 *otherwise. In both cases, the FlipFlop algorithm is optimal.*

Proof. When there is only one group, $nchild = g$, it is clear that the FlipFlop algorithm requires exactly $2n_1 - 1$ slots, that is the lower bound. Hence, the FlipFlop algorithm is optimal.
If there are two groups, $g + 1 \leq nchild \leq 2g$, the FlipFlop algorithm schedules in the odd slots the sink children of the first group and in the even slots the sink children of the second group. We distinguish two cases:

- if the size of the $(g+1)^{th}$ subtree is identical to the size of the first subtree, the FlipFlop algorithm requires an additional slot to complete the schedule of the second group. $2n_1$ slots are used, that is the optimal number.
- otherwise, the FlipFlop algorithm does not need any additional slots. Hence, it uses $2n_1 - 1$ slots for its schedule.

In both cases, the FlipFlop algorithm is optimal. □

Theorem 5. *In a multichannel WSN, the FlipFlop algorithm is not optimal for 3 groups.*

Proof. We just point out an example where the FlipFlop algorithm is not optimal. We consider a multiline topology with 16 nodes, 2 sink interfaces, 2 channels and 5 sink children. Hence, $g = min(5, 2, 2) = 2$. The topology is depicted in Figure 4.4. We notice that the third group contains only one subtree rooted at a sink child. The FlipFlop algorithm needs 9 slots, whereas the optimal slot number is $8 = \lceil \frac{N-1}{g} \rceil$. This can be explained by the fact that there are 3 $slots > g = 2$ where only the last sink child transmits. □

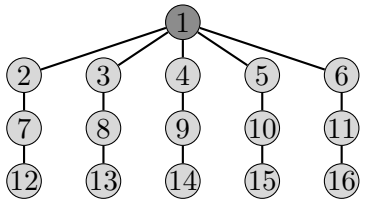


Figure 4.4: An example of topology with 16 nodes and 2 sink interfaces where FlipFlop is not optimal.

Theorem 6. *In a multichannel WSN, the optimal number of slots for a raw data convergecast is:* $max(\lceil \frac{N-1}{g} \rceil, 2n_1 - 1 + \delta)$, *if* $nchild > 2g$.

Proof. We now consider the case $nchild > 2g$ and prove by induction that there exists an algorithm that uses $n_{slot} = max(2n_1 - 1 + \delta, \lceil \frac{N-1}{g} \rceil)$. This is true for $nchild = N - 1$, in this case the sink has $nchild$ children which are leaves. Each sink child has exactly one message to transmit. In each slot, the algorithm schedules a group. Consequently, the number of slots needed is $n_{slot} = \lceil \frac{N-1}{g} \rceil$. Since $n_1 = 1$, we have $max(2n_1 - 1 + \delta, \lceil \frac{N-1}{g} \rceil) = \lceil \frac{N-1}{g} \rceil$.
Let us consider any topology with N nodes and $nchild > 2g$. From this topology, we can build a topology with $N - 1$ nodes by removing a node in the last group, while maintaining the non increasing order of the number of descendants in the subtrees. According to our

induction assumption, there exists an optimal algorithm that schedules any topology with $N-1$ nodes and $nchild \geq 2g$ in $n_{slot} = max(2n_1 - 1 + \delta, \lceil \frac{N-2}{g} \rceil)$. With regard to the scheduling of the topology with $N-1$ nodes, the node inserted and all its ascendants require the transmission of an additional message. The transmissions corresponding to this message, except the transmission by the root of this subtree, do not require any sink interface, we schedule them at the earliest (i.e. in the first slot where it is possible), in parallel with other nodes, without requiring any additional slot. It follows that the only transmission that remains to be scheduled is the transmission done by the root of the subtree involved. Let *child* denote this node. We consider two cases:

- $2n_1 - 1 + \delta \geq \lceil \frac{N-1}{g} \rceil$: the schedule length is imposed by the two first groups.
  Since the associated topology with $N-1$ nodes is obtained by removing a node in the last group and the number of groups is strictly higher than two, we also have $2n_1 - 1 + \delta \geq \lceil \frac{N-2}{g} \rceil$. According to our induction assumption, there exists an algorithm that reaches this bound of $2n_1 - 1 + \delta$. We modify this schedule to insert the additional transmissions required by the insertion of a node, as explained previously. The transmission of *child* can be scheduled in the last uncomplete slot where *child* is not transmitting. Such a slot exists, since *child* does not belong to the two first groups that impose the schedule length and the message originated from the inserted node has already reached *child*.

- $\lceil \frac{N-1}{g} \rceil > 2n_1 - 1 + \delta$.
  Since the associated topology with $N-1$ nodes is obtained by removing a node in the last group and the number of groups is strictly higher than two, we also have $\lceil \frac{N-2}{g} \rceil \geq 2n_1 - 1 + \delta$. According to our induction assumption, there exists an algorithm that reaches this bound of $\lceil \frac{N-2}{g} \rceil$. We modify this schedule to insert the additional transmissions required by the insertion of a node, as explained previously. For the transmission of *child*, we consider two cases:

  - $\lceil \frac{N-2}{g} \rceil = \lceil \frac{N-1}{g} \rceil$, there are uncomplete slots in the schedule of the $N-1$ topology. We distinguish again two subcases:

    * If in the last incomplete slot, *child* is not transmitting then the transmission of *child* can be scheduled in this slot. Notice that the message originated from the inserted node has already reached *child*. Hence, the number of slots required for the N nodes and the $N-1$ nodes topologies are identical.
    * Otherwise, we go backward to find a slot such that a sink child *other* $\neq$ *child* that does not transmit in the last slot is transmitting and no child of *child* is transmitting. We then exchange the transmissions of *child* and *other*. Consequently, we have found a schedule for the N node topology with the same number of slots than the $N-1$ topology, such that all messages sent by the new inserted node are transmitted to the sink.

  - $\lceil \frac{N-2}{g} \rceil = \lceil \frac{N-1}{g} \rceil - 1$, there is no slot in the schedule of the $N-1$ topology where the transmission of this sink child can be done. Hence, an additional slot is inserted at the end of the schedule.

Consequently, we have established a schedule for the N topology that reaches the bounds. Hence, the theorem. □

4.4 Theoretical bounds for heterogeneous traffic

We generalize here our theoretical results provided in the last section giving minimal bounds for raw data convergecast, taking into account both heterogeneous node demands and the acknowledgment policy.

4.4.1 Without immediate acknowledgment

Lemma 3. *In any WSN with heterogeneous demands of nodes, a lower bound on the number of slots required by a raw data convergecast is S_n.*

Proof. In any network with heterogeneous node demands, the sink has to receive $\sum_{u \neq sink} Gen(u)$ messages from its children. The number of simultaneous transmissions to the sink is limited by the number of children, the number of sink interfaces, as well as the number of available channels (each interface using a channel different from the channels used by the other active interfaces). Hence, the number of slots needed is higher than or equal to $\lceil \frac{\sum_{u \neq sink} Gen(u)}{min(nchild, nchannel, ninterf)} \rceil$; hence, the lemma. □

4.4.1.1 Linear Networks

Lemma 4. *In any linear network with heterogeneous demands of nodes, where each node has $nchannel > 1$, the minimum number of slots for a raw data convergecast is $Gen(ch1) + 2\sum_{u \neq sink, u \neq ch1} Gen(u)$, whatever the number of interfaces that the sink has, where $Gen(u)$ is the number of slots needed by node u, to transmit its own data to its parent.*

Proof. Consider any linear network with heterogeneous demands of nodes. The sink has only one child which corresponds to $ch1$. This latter needs to transmit $\sum_{u \neq sink} Gen(u)$ packets to the sink and needs to receive $\sum_{u \neq sink, u \neq ch1} Gen(u)$ packets from its child. Since node $ch1$ has a single interface, these transmissions cannot overlap. As a consequence, a lower bound on the number of slots is equal to $Gen(ch1) + 2\sum_{u \neq sink, u \neq ch1} Gen(u)$; hence, the Lemma. □

Lemma 5. *In any linear network with heterogeneous demands of nodes, where each node has $nchannel > 1$, the algorithm that schedules:*

- *In the current time slot, $t \leq 2 * \sum_{u \neq sink, u \neq ch1} Gen(u)$,*
  - *if t is odd, any node that is $(2h+1)$ hops away from the sink, with $h \in [0, \lfloor \frac{N-1}{2} \rfloor]$, where N is the number of nodes;*
  - *if t is even, any node that is $2h$ hops away from the sink, with $h \in [1, \lfloor \frac{N-1}{2} \rfloor]$.*
- *the child of the sink, $ch1$, in the next contiguous $Gen(ch1)$ slots,*

provides the minimum number of slots for a raw data convergecast.

Proof. According to the algorithm described above,

- In the first slot, the sink child transmits one packet of its own data.
- In any odd slot, t, with $1 < t \leq 2 * \sum_{u \neq sink, u \neq ch1} Gen(u)$, the sink child transmits any packet received from its children in the previous slot. Consequently this schedule ensures that any node, different from the sink and its child, transmits simultaneously with its grand-parent. Since these two nodes are conflicting on the same channel, they are only allowed to transmit on different channels. For instance, if we consider the odd slots, nodes at depth $4h+1$ will transmit on the first available channel, whereas nodes at depth $4h+3$ will transmit on the second available channel. For the even slots, nodes at depth $4h$ will transmit on the first available channels, and nodes at depth $4h+2$ will transmit on the second available channel.
- In the $(2 * \sum_{u \neq sink, u \neq ch1} Gen(u) + 1)^{th}$ slot, the sink child transmits the last last packet received from its child.
- In the next $(2 * \sum_{u \neq sink, u \neq ch1} Gen(u) - 1)$ slots, it transmits its $\sum_{u \neq sink, u \neq ch1} Gen(u) - 1$ own packets.

With this schedule, any node alternates transmit and receive slots until it has transmitted its own data and the data generated by its downstream nodes. Hence, the child of the sink is kept busy in all slots, leading to a number of slots equal to this given by Lemma 4; hence, the lemma. □

4.4.1.2 Multi-Line Networks

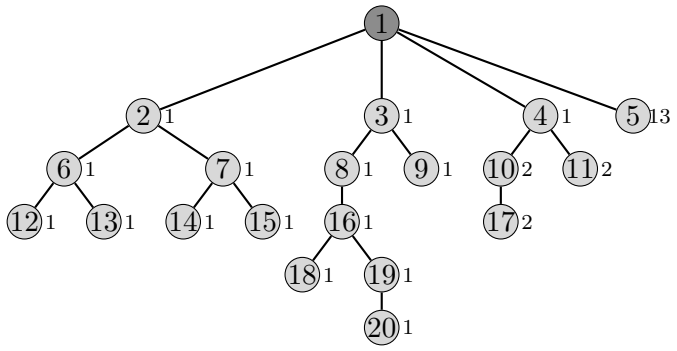
Theorem 7. *In any multi-line network with heterogeneous demands of nodes, a lower bound on the number of slots for a raw convergecast is Max(S_n,S_t).*

Proof. The sink requires at least $\lceil \frac{\sum_{u \neq sink} Gen(u)}{min(nchild, nchannel, ninterf)} \rceil$ time slots to receive all the packets generated in the network during one cycle (see Lemma 3). Moreover, let us consider the subline rooted at any sink child, i. From Lemma 4, at least $\sum_{u \neq i,\ u \in subline(i)} Gen(u)$ slots are required to receive data and $\sum_{u \in subline(i)} Gen(u)$ slots to transmit data to the sink. Furthermore, let us order the sink children according to the decreasing order of the number of slots they need, that is, $Gen(i) + 2 \sum_{v \neq i, v \in subline(i)} Gen(v)$ for the sink child i. If the $(g+1)^{th}$ sink child requires the same number of slots as the first one, then its schedule will require an additional slot. Indeed, the schedule of this line requires the same number of slots as the first one. However, the sink child starts to transmit one slot later, because there is no available channel or interface. Consequently, it will end one slot after. Hence, the value of δ. Of course, the number of slots should be dimensioned to meet the strongest requirements of the subline; hence, the theorem. □

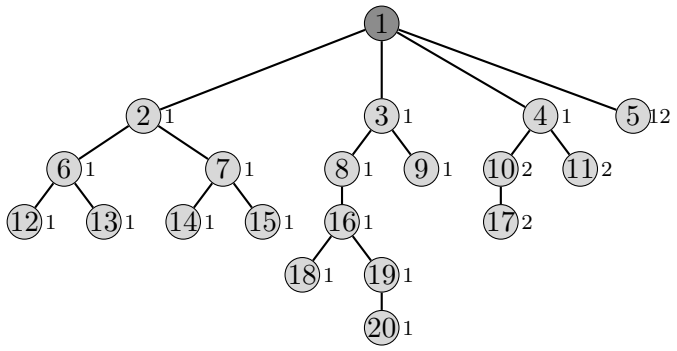
4.4.1.3 Tree Networks

Theorem 8. *In tree networks with heterogeneous demands of nodes, a lower bound on the number of slots for a raw data convergecast is Max(S_n,S_t).*

Proof. The sink requires $\lceil \frac{\sum_{u \neq sink} Gen(u)}{g} \rceil$ time slots to receive all the packets generated in the network, as seen in Lemma 3. Furthermore, each child i of the sink has $\sum_{v \neq i \in subtree(i)} Gen(v)$ packets to receive and $\sum_{v \in subtree(i)} Gen(v)$ packets to transmit. Moreover, let us order the sink's children according to the decreasing order of the number of slots they need, that is, $Gen(i) + 2\sum_{v \neq i, v \in subtree(i)} Gen(v)$ for the sink child, i. If the $(g+1)^{th}$ sink child requires the same number of slots as the first one, then its schedule will require an additional slot. Indeed, the schedule of this subtree requires the same number of slots as the first one. However, the $(g+1)^{th}$ sink child starts to transmit one slot later than the first one, because at the first slot, all the available interfaces of the sink or all the available channels are used by the g first children of the sink. Consequently, the $(g+1)^{th}$ sink child will end one slot after; hence the value of δ. Of course, the minimum number of slots should work for the subtree requiring the highest number of slots; hence the theorem. □



(a)



(b)

Figure 4.5: Case where $\delta = 1$ when the sink is equipped with three radio interfaces and three channels. (**a**) (3 channels;3 radio interfaces) = 14 slots; (**b**) (3 channels;3 radio interfaces) = 13 slots

An example justifying the existence of δ in the lower bound for tree topologies is given in Figure 4.5. The sink is equipped with three radio interfaces and three channels are available. So $g = min(nchild, nchannel, ninterf) = min(5, 3, 3) = 3$. In the first tree, as depicted in Figure 4.5(a), the first group contains the three subtrees rooted respectively at node 2, 3

and 4. The second group contains a single subtree rooted at node 5. This latter requires 13 slots to send its own data. Besides, node 2, which is the most demanding node in the first subtree, needs also 13 slots to send it own data and the data of its descendants. Hence, to complete convergecast, 14 slots are mandatory ($\delta = 1$). Nevertheless, in the second tree, as shown in Figure 4.5(b), node 5 needs a number of slots smaller than the number needed by node 2. Thus, the lower bound of schedule length is dictated by node 2. Thus, only 13 slots are required.

4.4.1.4 T_t and T_n Configurations

By definition, a configuration is given by a topology (set of nodes and links) and the initial demands of nodes.

In the case of multiline and tree networks, we define two types of configurations, T_t and T_n:

- A configuration is denoted T_t when the optimal number of slots is imposed by the most demanding subtree rooted at a sink child, i. Its demand is equal to $Gen(i) + 2\sum_{v \neq i, v \in subtree(i)} Gen(v)$.
- A configuration is denoted T_n when the optimal number of slots depends only on the total number of demands and $g = min(nchild, nchannel, ninterf)$. It is equal to $\lceil \frac{\sum_{u \neq sink} Gen(u)}{g} \rceil$. Notice that a T_n configuration corresponds to a Capacitated Minimal Spanning Tree, where each branch has a total demand for slots, $\leq \lceil \frac{\sum_{u \neq sink} Gen(u)}{g} \rceil$ [Papadimitriou 1978].

We now give some additional definitions:

- the T_n $configurations$ are traffic-balanced. Hence, their number of slots is given by S_n.
- the T_t $configurations$ are dominated by the subtree requiring the highest number of transmissions. Hence, their number of slots is given by S_t.
- A subtree is said active in a slot t if and only if the node at which it is rooted is either transmitting or receiving in this slot.
- A star topology is a tree topology of depth 1.

4.4.2 With immediate acknowledgment

Lemma 6. *For a raw data gathering in a multichannel WSN, the only additional conflicts created by the choice of the immediate acknowledgment are those occurring between a node and one of its nephews (i.e. a child of a brother of the node considered).*

Proof. The additional conflicts introduced by the immediate acknowledgment policy are those illustrated by Figures 4.6(7) and 4.6(8). We distinguish two cases:

- First case: conflict illustrated by Figure 4.6(7): Since in a data gathering tree, any node different from the sink transmits only to its parent in the tree and since there is no radio link other than those present in the data gathering tree (Assumption 6), this case can occur only when Tx_1 is a node u different from the sink, Rx_1 is the parent of u, whereas Rx_2 is a brother of u and Tx_2 is a child of Rx_2.

- Second case: conflict illustrated by Figure 4.6(8): Such a conflict can never occur, since in the tree no two one-hop neighbors can transmit simultaneously to two different parents. Hence, the only additional conflicts are those occurring between a node different from the sink and one of its nephews as depicted in Figure 4.7.

□

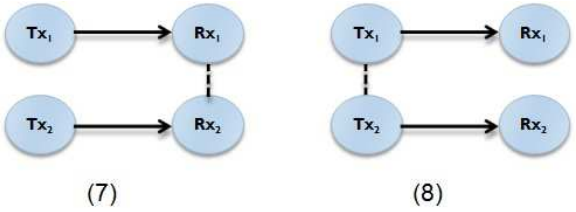


Figure 4.6: Additional conflicting transmissions with immediate acknowledgment.

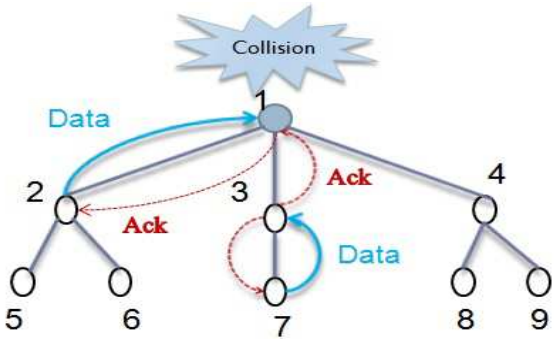


Figure 4.7: Collision between a node and its nephew.

Lemma 7. *The minimum number of slots required for a raw data gathering in a multichannel WSN having a star or line topology, whatever the acknowledgment policy, is equal to* $max(S_n, S_t)$.

Proof. A star topology or a line topology does not create any conflict between a node and its nephew. Hence, the minimum number of slots does not depend on the acknowledgment policy. From Theorem 8, it is equal to $max(S_n, S_t)$. □

Theorem 9. *When* $nchild \leq min(nchannel, ninterf)$ *or* $ninterf < min(nchild, nchannel)$*, the minimum number of slots required for a raw data gathering in a multichannel WSN generating heterogeneous traffic is the same with and without the immediate acknowledgment.*

Proof. We distinguish two cases.

• First case: If $nchild \leq min(nchannel, ninterf)$: We can build a schedule where the subtree rooted at any sink child ch is active in any slot t with $1 \leq t \leq Trans(ch)$ and nodes are scheduled according to their depth in the tree. Each child of the sink receives both its own interface and its own channel. In each brotherhood of depth $4h+1$, with h integer ≥ 1, one node u (the selection policy of this node in its brotherhood is round robin) receives the same odd slot t and the same channel as the root of its subtree (if u has still packets to transmit). Similarly, in each brotherhood of depth $4h+3$, one node receives the same odd slot as the root of its subtree (if this slot is needed) but with a different channel (e.g. the channel of the root of the previous subtree). Alternatively in the even slots and as long as they have packets to transmit, in each brotherhood of even depth, one node is scheduled on the channel used by its parent in the previous slot. Since in the absence of acknowledgment, this schedule provides the minimum number of slots required by the most demanding subtree, given by S_t, this schedule is optimal. Since there is no slot where a node and its nephew are scheduled in the same slot and on the same channel, according to Lemma 6, this schedule is also valid if the immediate acknowledgment is used. Hence, the first part of the theorem.

• Second case: If $ninterf < min(nchild, nchannel)$: At any slot at most $min(nchannel, nchild, ninterf) = ninterf$ children of the sink are scheduled. Let us consider any valid schedule without acknowledgment that provides the minimum number of slots and build a valid schedule for the immediate acknowledgment. We distinguish two cases:

◦ First subcase: there is no slot where a node and its nephew transmit in the same slot and on the same channel. Hence, according to Lemma 6, this schedule is also valid if the immediate acknowledgment is used.

◦ Second subcase: there is a slot where a node and its nephew, denoted n, transmit in the same slot and on the same channel. We focus on t the smallest time slot where this occurs. Since $nchannel > ninterf$, at most $ninterf$ uncles of n are scheduled in the same slot but each on its own channel. It is necessary to schedule the transmission of node n on a channel c that is unused by the uncles of n scheduled in this slot. Such a channel exists since $nchannel > ninterf$. Furthermore, we have to avoid the creation of new conflicts: for instance a conflict between n and ch, a child of a child of n, when ch is scheduled in the same slot and on the same channel c as n. We then apply the following rules: any descendant of n at a depth $= 4h+4$ with h integer ≥ 0 is scheduled on a channel already used by a child of the sink, whereas any descendant of n at a depth $= 4h+6$ with h integer ≥ 0 is scheduled on the channel c. Hence, the conflict caused by n or any of its descendant is avoided. This method can be applied to any conflicting node n. Finally, we get a valid schedule using the same number of

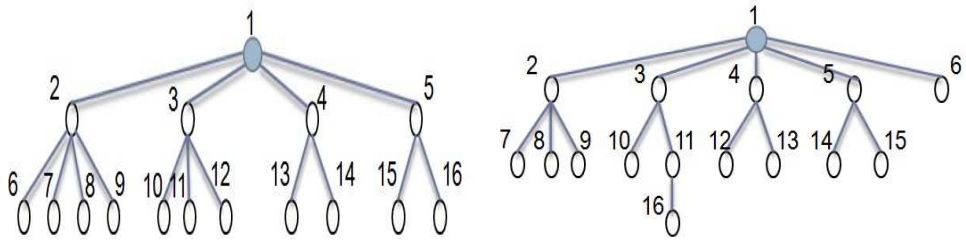
slots as the initial one. Hence, the immediate acknowledgment does not require any additional slot. Hence, the second part of the theorem.

□

Theorem 10. *If* $nchannel \leq ninterf < nchild$ *and* $g \times max(S_t, S_n) \geq \sum_{u \in\ WSN} Gen(u) + Rcv(ch))$*, the minimum number of slots required for a raw data gathering in a multichannel WSN generating heterogeneous traffic is equal to* $max(S_t, S_n)$*, with and without the immediate acknowledgment, where* $Rcv(ch)$ *denotes the number of packets received by ch, the most receiving child of the sink.*

Proof. The quantity $g \times max(S_t, S_n)$ denotes the number of transmission opportunities available for the children of the sink. The number of messages that must be transmitted to the sink is equal to $\sum_{u \in\ WSN} Gen(u)$. Each of this transmission uses an interface and a channel among the g available interfaces and channels. If $nchannel \leq ninterf < nchild$, the only possibility for any child of a sink child is to transmit on a channel selected among the g channels. Hence, with the immediate acknowledgment policy, a conflict would occur with the child of the sink. To avoid such a conflict, we should have: $g \times max(S_t, S_n) \geq \sum_{u \in\ WSN} Gen(u) + Rcv(ch))$, where ch denotes the most receiving child of the sink. In such a case, the conflict is avoided and the number of slots is the same with and without immediate acknowledgment; it is equal to $max(S_n, S_t)$. Hence, the theorem. □

We now prove the existence of T_t and T_n configurations that both require different numbers of slots, depending on the acknowledgment policy. We assume 2 interfaces and 2 channels. We have $nchannel \leq ninterf < nchild$ in both configurations. For the T_t configuration (see Figure 4.8(a)), we have $S_t = 9 > S_n = 8$, 9 slots are needed without acknowledgment and 10 slots with immediate acknowledgment. For the T_n configuration (see Figure 4.8(b)), we have $S_n = 8 > S_t = 7$, 8 slots are needed without acknowledgment and 9 slots with immediate acknowledgment. We notice that neither T_t nor T_n meets $g \times max(S_t, S_n) \geq \sum_{u \in\ WSN} Gen(u) + Rcv(ch))$, explaining why the number of slots differs according to the acknowledgment policy.



(a) in T_t configuration (b) in T_n configurations

Figure 4.8: configuration whose number of slots differs with/without immediate acknowledgment.

4.5 Conclusion

This chapter is concerned with the delay in collecting data from sensor nodes to the sink. We consider applications wherein data packets generated by every sensor have to reach the sink. The minimum number of slots for raw data convergecast was computed in line and trees topologies and it is respectively $2N-3$ and $\text{Max}(S_n^1,S_t^1)$. Corresponding optimal scheduling strategies were also described. We focused our analysis on networks where nodes generate heterogeneous traffic toward the sink which can be equipped by multiple radio interfaces. Immediate acknowledgment was also considered and specific configurations, where immediate acknowledgement does not alter the optimal bounds, were determined. Besides, we evaluated the tradeoff between the number of channels and scheduling length and gave guidelines on the choice of number of channels. Indeed, We have shown that for a sink with a single interface, it is useless to have a number of channels higher than two, since the optimal number of slots is already reached with two channels. For a sink with $ninterf$ multiple radio interfaces, $ninterf > 1$, $ninterf$ channels are sufficient to get the optimal schedule length.

In the next chapter, we are interested in determining a TDMA schedule such that the entire convergecast can be achieved in the minimal number of time slots. The theoretical study done in this chapter will help us to evaluate how close our designed algorithm is to the established theoretical lower bounds.

Chapter 5

MODESA: an Optimized Multichannel Slot Assignment for Raw Convergecast in WSNs

Contents

5.1 Introduction . . . 65
5.2 State of the art . . . 66
5.3 Multichannel slot assignment problem . . . 67
5.3.1 Formalization of the problem . . . 67
5.3.2 Illustrative examples . . . 69
5.4 MODESA: Multichannel Optimized DElay time Slot Assignment . 70
5.4.1 Principles . . . 70
5.4.2 The MODESA algorithm . . . 71
5.4.3 Optimality of MODESA for homogeneous traffic . . . 73
5.5 Performance evaluation of MODESA . . . 75
5.5.1 MODESA with homogeneous traffic . . . 75
5.5.2 MODESA with heterogeneous traffic . . . 80
5.5.3 Impact of additional links . . . 82
5.6 MODESA improvement . . . 84
5.6.1 Channel allocation strategy . . . 84
5.6.2 Multipath transmission scheduling . . . 86
5.6.3 Different topologies on different channels . . . 89
5.7 Conclusion . . . 92

5.1 Introduction

WSNs have been frequently put into use for data gathering applications, where nodes send their sensed data to a sink over a routing tree. Such a communication scheme is called con-

vergecast. In these applications, if every packet is forwarded individually, we speak about *raw-data convergecast*. In such a case, intermediate nodes in the routing tree simply apply the store and forward strategy, without processing the received packets. Most of these applications share the requirement of deterministic delay bounds and guaranteed on packet delivery. The delay requirement of these applications may be difficult to meet with a single wireless channel.

As seen in chapter 3 and chapter 4, the multichannel paradigm was proposed to improve network throughput, reduce packet loss and minimize the time needed to complete convergecast in multichannel WSNs.

The new standard $IEEE802.15.4e$ proposed the TimeSlotted Channel Hopping (TSCH) [TG4e 2012] mode where nodes perform channel hopping. This latter, combined with a centrally built slotted schedule, ensures collision-free communications and high reliability against interferences. However, the standard does not propose a mechanism to built this schedule. That is why, in this chapter, we cover this gap by proposing a conflict-free multichannel time slot assignment that minimizes the data gathering delay. Moreover, we are interested in a joint channel and time slot assignment that is traffic aware: each node is assigned a number of slots equal to its traffic demand. Indeed, nodes close to the root of the convergecast tree have a higher traffic load.

After a state of the art in Section 5.2, we first formalize the problem as a linear program in Section 5.3 and determine the minimum number of slots for various multichannel topologies using GLPK solver. We then propose our joint time slot and channel assignment algorithm called MODESA in Section 5.4. We prove its optimality for different topologies and present its variants achieving a load balancing between the channels used in Section 5.6. Performances of MODESA are evaluated by simulation in Section 5.5. Finally, we conclude in Section 5.7.

5.2 State of the art

We detail in this section TDMA based scheduling protocols proposed in multichannel WSNs that minimize data gathering delays . WirelessHART [Song 2008] was the first communication standard specially designed to fit critical requirements of industrial applications. WirelessHART uses TDMA to arbitrate communications between devices. To enhance reliability, TDMA is combined with channel hopping on a per-transaction (packet + acknowledgment) basis. A fundamental shortcoming of this standard is that only a single device is scheduled for transmission on each channel in the same slot. Hence, there is no spatial reuse of the bandwidth.

In [Zhang 2009], authors address jointly the link scheduling and channel assignment for convergecast in networks operating according to the WirelessHART standard. Authors have proven that for linear networks with N single buffer devices, the minimum schedule length obtained is $(2N-1)$ time slots with $\lceil N/2 \rceil$ channels. They present also an algorithm with time complexity $O(N^2)$ to generate a time and channel optimal convergecast schedule. The solution does not provide spatial reuse of the bandwith and is restricted to linear topologies which are not suitable for all real deployments. In addition, they focus only on single radio interface devices.

TMCP [Wu 2008] was designed to support data collection traffic. It begins by partitioning the network into multiple subtrees and then assigns different channels to nodes belonging to different subtrees. Hence, it minimizes the interferences between subtrees. After the channel assignments, time slots are assigned to nodes. However, TMCP does not eliminate contention inside the branches of a subtree since nodes that belong to the same branch communicate on the same channel.

Y-MAC [Kim 2008] is a multichannel MAC protocol for WSNs that requires that nodes share the same wake/sleep duty cycle. Time slots are not assigned to the senders but to the receivers. At the beginning of each time slot, potential senders for the same receiver contend for the medium. When a node needs to transmit multiple packets to a receiver, these packets are sent on different channels following a pre-determined hopping sequence. We notice that Y-MAC has not been designed for data gathering applications, where the contention around the sink quickly becomes severe especially in heavy traffic conditions.

In [Raman 2010], Raman et al have proposed PIP : a joint TDMA-FDMA based bulk transfer protocol. When the sink needs data from a specific sensor, it establishes a connection with this latter and downloads data from that node at the highest rate possible. The major shortcoming of PIP is that the sink can only collect data from at most two sensors at the same time (i.e at most two simultaneous connections).

Incel et all [Incel 2012], have proven that if all interfering links are removed (with the required number of channels), the schedule length for raw-data convergecast is lower bounded by $max(2n_k - 1, N)$ where n_k is the maximum number of nodes in any top-subtree of the routing tree and N is the number of source nodes. They have also proposed an optimal convergecast scheduling algorithm JFTSS that achieves this lower bound on any network topology where the routing tree has an equal number of nodes on each branch.

Authors of [Palattella 2012] proposed TASA, a centralized traffic-aware scheduling algorithm for 802.15.4*e* based networks. TASA proceeds in two steps:(1) matching step where links eligible to be scheduled in the same time slot are selected (2) coloring step where each link selected for transmission is assigned a channel offset. The channel offset is translated into a frequency using a translation function. However, TASA does not take into account queue congestion.

5.3 Multichannel slot assignment problem

We are looking for a multichannel slot assignment model that minimizes the number of slots assigned, under the assumptions 1, 2, 3, 5 and 6 given in the previous chapter and ensures that no two conflicting nodes transmit simultaneously on the same channel. The assumption 6 will be relaxed in section 5.5.3 to evaluate the impact of additional interfering links on schedule length.

5.3.1 Formalization of the problem

The network is formalized as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices representing network nodes and $\mathcal{E}$ is the set of edges representing the communication links. Let $\mathcal{V} =$

$\mathcal{V}_s \bigcup \mathcal{V}_g$, where $\mathcal{V}_s$ is the set of source nodes in the network and $\mathcal{V}_g$ represents the set of gateways acting as sinks, with $\mathcal{V}_s \bigcap \mathcal{V}_g = \emptyset$.

For each node $v \in \mathcal{V}$, we define $Confict(v)$ the set of nodes that interfere with v when transmitting on the same channel. Moreover, for any source s, let $Gen(s)$ be the number of packets generated by s that it has to transmit in the TDMA cycle. For any link e, let $f_{e,s}$ denote the number of packets generated by the source s and sent over the link e during the TDMA cycle. Let $\mathcal{E}^+(v)$ denote the set of links through which a node v can transmit. Let $\mathcal{E}^-(v)$ be the set of links through which a node v can receive.

Let $\mathcal{C}$ be the set of the *nchannel* channels available for any transmission. We define $a_{e,c,t}$ the activity of a link e on the channel c in the time slot t, ie $a_{e,c,t} = 1$ if and only if there is a transmission of a packet on the link e on the channel c in the time slot t and $a_{e,c,t} = 0$ otherwise. Furthermore, let u_t be the use of a slot t, in other words $u_t = 1$ means that there is at least one link activity on any channel in the slot t and $u_t = 0$ denotes an empty slot.

We can compute T_{max}, an upper bound of the cycle length. This bound is reached when all packets are sent sequentially on the same channel. We then have: $T_{max} = \sum_s \sum_e f_{e,s} * depth_s$ where $depth_s$ is the depth of node s in the the data gathering tree. The objective is to minimize the number of slots $t \leq T_{max}$. Using the variable u_t, the minimization ensures there are no idle slots.

$$min \sum_{t \leq T_{max}} u_t$$

with the following constraints:

$$\begin{aligned} & a_{e,c,t} \leq u_t \\ & \quad \forall e \in \mathcal{E}, \forall c \in \mathcal{C}, t \leq T_{max} \end{aligned} \tag{5.1}$$

$$\begin{aligned} & a_{e,c,t} + a_{e',c,t} \leq 1 \\ & \quad \forall v \in \mathcal{V}, \forall e \in \mathcal{E}^+(v), \\ & \quad \forall w \in Conflict(v), \forall e' \in \mathcal{E}^+(w), \\ & \quad \forall c \in \mathcal{C}, t \leq T_{max} \end{aligned} \tag{5.2}$$

$$\begin{aligned} & \sum_{c \in \mathcal{C}} \sum_{e \in \mathcal{E}^+(v)} a_{e,c,t} + \sum_{c \in \mathcal{C}} \sum_{e' \in \mathcal{E}^-(v)} a_{e',c,t} \leq 1 \\ & \quad \forall v \in \mathcal{V}, t \leq T_{max} \end{aligned} \tag{5.3}$$

$$\begin{aligned} & \sum_{s \in \mathcal{V}_s} f_{e,s} = \sum_{c \in \mathcal{C}} \sum_{t \leq T_{max}} a_{e,c,t} \\ & \quad \forall e \in \mathcal{E} \end{aligned} \tag{5.4}$$

$$\begin{aligned} & \sum_{e \in \mathcal{E}^+(s)} f_{e,s} = Gen(s) \\ & \forall s \in \mathcal{V}_s \end{aligned} \tag{5.5}$$

$$\begin{aligned} & \sum_{g \in \mathcal{V}_g} \sum_{e \in \mathcal{E}^-(g)} f_{e,s} = Gen(s) \\ & \quad \forall s \in \mathcal{V}_s \end{aligned} \tag{5.6}$$

$$\sum_{s\in\mathcal{V}_s}\sum_{e\in\mathcal{E}^+(i)} f_{e,s} = Gen(i) + \sum_{s\in\mathcal{V}_s}\sum_{e\in\mathcal{E}^-(i)} f_{e,s} \quad \forall i \in \mathcal{V}_s \tag{5.7}$$

$$\begin{aligned}\sum_{c\in\mathcal{C}}\sum_{e\in\mathcal{E}^+(i)} a_{e,c,t} \leq & \sum_{c\in\mathcal{C}}\sum_{e\in\mathcal{E}^-(i)}\sum_{t'\in\{1..t-1\}} a_{e,c,t'} + Gen(i) \\ & - \sum_{c\in\mathcal{C}}\sum_{e\in\mathcal{E}^+(i)}\sum_{t'\in\{1..t-1\}} a_{e,c,t'} \\ & \forall i \in \mathcal{V}_s, t \leq T_{max}\end{aligned} \tag{5.8}$$

Constraint 5.1 binds the use of a time slot to at least the activity of one link on any channel in the slot. Constraint 5.2 ensures that two conflicting nodes do not transmit on the same channel in the same time slot. Constraint 5.3 guarantees that the number of simultaneous communications for a node is limited to its number of interfaces. Constraint 5.4 ensures the mapping between the activities on all channels and the packets sent on links. Constraints 5.5, 5.6 and 5.7 express the conservation of messages between the sources and the sinks. The last constraint guarantees that any node receives or generates a packet before transmitting it.

The above problem can be solved using an ILP solver such as GLPK [GLPK 2011]. The next section will outline the solutions obtained by GLPK on typical multichannel WSN topologies.

5.3.2 Illustrative examples

An optimal multichannel time slot assignment can be obtained by linear programming tools such as GLPK (GNU Linear Programming Kit) [GLPK 2011] based on the described model in previous section. Figure 5.1 shows the optimal number of slots for different multichannel topologies (linear Figure 5.1(a), multiline Figure 5.1(b), balanced tree Figure 5.1(c) and tree Figure 5.1(d)) with various number of sink interfaces $ninterf$ and channels $nchannel$. To simplify the computation, we assume that each node generates exactly one packet at each round of convergecast. These optimal results are reached by the MODESA algorithm presented in Section 5.4.

The obtained results confirm the theoretical results computed in the previous chapter. Indeed, we distinguish two types of network topologies T_t and T_n where the optimal number:

- is imposed by the most populated subtree: see for instance Figures 5.1(b) and 5.1(d), both with two sink interfaces. In Figure 5.1(d), there are two most populated subtrees rooted at nodes 2 and 3 respectively.
- depends only on the number of nodes and the number of interfaces of the sink: see for instance Figures 5.1(c) and 5.1(d) both with a single sink interface.

We can observe also that a given topology may belong to one type or another depending on the number of sink interfaces (e.g. Figures 5.1(c) and 5.1(d) for a single sink interface belong to the first type and for two sink interfaces to the second type).

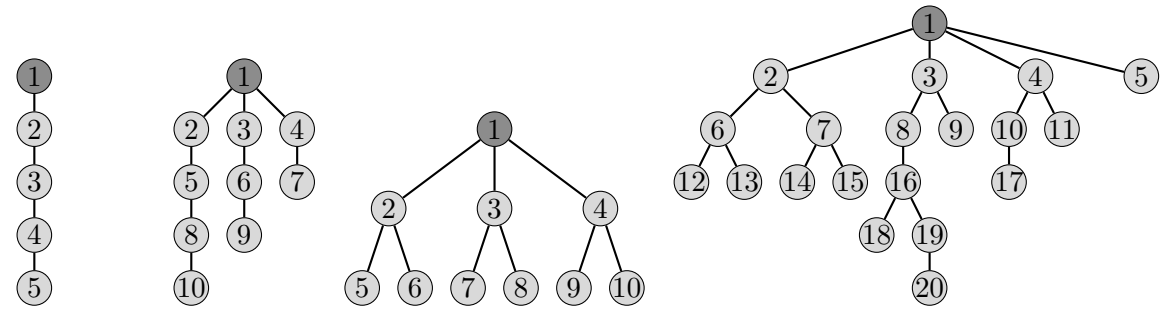


Figure 5.1: The optimal number of slots, *nbOptimalSlots*, for various topologies with different numbers of sink interfaces $ninterf$ and channels *nchannel* with the notation: ($ninterf$;*nchannel*)=*nbOptimalSlots*.

Computing the optimal schedule still need to solve the ILP equations on the sink. Moreover, solving this integer program is an NP-Hard problem. Therefore, it is carried out infrequently in small size networks. To schedule larger networks or adapt to traffic changes, we have to rely on approximation solutions or suboptimal heuristics that do no guarantee the optimality of the provided solution. Thus, we design an efficient heuristic algorithm which does not solve the ILP equations. This is the purpose of the next section.

5.4 MODESA: Multichannel Optimized DElay time Slot Assignment

The aim of this section is to propose a centralized raw data convergecast scheduling, called MODESA, that takes into account the availability of multiple channels to reduce the TDMA cycle length while ensuring a medium access opportunities proportional to the traffic demand of the node considered.

5.4.1 Principles

MODESA builds the multichannel scheduling slot by slot applying the following rules:

1. Any node u has a *dynamic priority*. The priority is equal to $remPckt(u) * Rcv(Parent(u))$ where $remPckt(u)$ is the number of packets the node has in its buffer at the current iteration. $Rcv(Parent(u))$ is the total number of packets the parent of the node has to receive in a cycle. The idea behind this heuristic is to reduce the number of buffered packets by favoring nodes having packets to transmit to a parent with a high number of descendants.

2. Nodes compete for the current time slot if and only if they have data to transmit.

3. In addition to be allowed to transmit in a slot, a node and its parent must have an available interface.

4. For any slot, the first node scheduled is the node having the highest priority among all the nodes having data to transmit. If two nodes have the same priority, MODESA chooses the node with the smallest identifier. The node selected is scheduled on the first channel c in the greedy variant of MODESA.

5. Any node obeying rules 2 and 3 is scheduled in the current time slot on channel c if it does not interfere with nodes already scheduled on channel c in this slot.

6. Conflicting nodes that interfere with nodes already scheduled in this slot are scheduled on a different channel.

5.4.2 The MODESA algorithm

MODESA pseudo-code is given by Algorithm 1. The algorithm iterates over N the set of nodes having data to transmit and sorted according to their priorities. In each iteration, the algorithm determines among these nodes the set of nodes that are assigned to the current time slot t. The node u with the highest priority that has an available interface as well as its parent is scheduled first. Then MODESA iterates on the set of nodes sorted according to their priority. Nodes which are non conflicting are allocated to the same channel. In contrast, any other node in the sorted set N is assigned the same time slot but on a different channel if it conflicts with nodes already assigned to the current slot.

Algorithm 1 MODESA algorithm with the greedy variant

Input: $nchannel$ channels, a spanning tree T, where each node u has i_u radio interfaces, $Gen(u)$ packets locally generated and a set of conflicting nodes $Conflict(u)$.
Output: The scheduling of nodes in the TDMA cycle
/* Initialization phase */
Initialize priority and traffic demand $Gen(u)$ for each node u
$t \leftarrow 0$ // current time slot
for each node u **do**
  $remPckt(u) \leftarrow Gen(u)$ // number of packets initially present in the buffer of u
end for
/* Scheduling phase */
while $\sum_u remPckt(u)$ **do** // there are packets to transmit
  Initialize number of available interfaces of nodes
  Initialize conflicting nodes on channel c, $conflict_c \leftarrow \emptyset, \forall c = 1..n_{channel}$
  $N \leftarrow$ list of nodes having data to transmit and sorted according to their priorities.
  $t \leftarrow t + 1$
  /* Assignment of slot t */
  while $N \neq \emptyset$ **do**
    $Tx \leftarrow False, nChannelReached \leftarrow False$
    repeat
      Select node v with the highest priority in N
      $N \leftarrow N \setminus \{v\}$
    until $i_v > 0$ and $i_{Parent(v)} > 0$ // v and its parent have an available interface
    $c \leftarrow 1$ // selected channel
    repeat
      if $v \notin conflict_c$ **then**
        Node v transmits in slot t on the channel c
        $remPckt(v) \leftarrow remPckt(v) - 1$
        $remPckt(Parent(v)) \leftarrow remPckt(Parent(v)) + 1$
        $i_v \leftarrow i_v - 1$
        $i_{Parent(v)} \leftarrow i_{Parent(v)} - 1$
        $conflict_c \leftarrow conflict_c \cup Conflict(v)$
        $Tx \leftarrow True$
      else
        if $c < nchannel$ **then**
          $c \leftarrow c + 1$ // change of selected channel
        else
          $nChannelReached \leftarrow true$
        end if
      end if
    until Tx || $nChannelReached$
  end while
  Update priority of nodes
end while

5.4.3 Optimality of MODESA for homogeneous traffic

In this section, we will prove the optimality of MODESA, our joint time slot and channel assignment algorithm, in some topologies. We consider homogeneous traffic where each node u has a $Gen(u) = 1$. A tree is said balanced if and only if at each level l, the number of children is the same for all nodes belonging to level l. We recall that the children of the sink, being ordered by decreasing order of their number of packets to transmit, the first group contains the $g = min(nchild, nchannel, ninterf)$ first subtrees having the highest demand. The second group contains the g next subtrees and so on.

Theorem 11. *When nchannel* > 1 *and traffic is homogeneous, MODESA is optimal for linear, multiline and balanced tree topologies.*

Proof. We consider three cases:

- First case: linear topologies
  The behavior of MODESA and FlipFlop, described in chapter 4 are identical: odd (even respectively) slots schedule the transmissions of nodes at an odd (even respectively) distance from the sink. Hence, MODESA is optimal for linear topologies.

- Second case: multiline topologies
  - When there is a single group, all sink children are scheduled in parallel. The number of slots is the one needed to schedule the first sink child. Hence, MODESA is optimal.
  - When there are two groups, MODESA schedules in the first slot the sink children of the first group because of their higher number of packets to receive and in the second slot the sink children of the second group. With two groups, a child of a sink child has never a priority higher than its parent, when this parent has at least one message to transmit. Hence, MODESA behaves exactly like FlipFlop. Since FlipFlop is optimal, MODESA is optimal too.
  - When there are more than two groups, we distinguish two cases:
    * If $2n_1 - 1 + \delta \geq \lceil \frac{N-1}{g} \rceil$, the most populated line determines the schedule length. Let $ch1$ be the sink child corresponding to this line. In any slot, MODESA schedules either $ch1$ or the child of $ch1$, keeping $ch1$ always active, either transmitting to the sink or receiving from its child. This gives the optimal schedule for this line. Furthermore, MODESA completes the schedule of any slot with the other sink children that have not yet received the slots they need, while keeping busy the g sink interfaces as long as possible. Hence, MODESA gets the optimal number of slots.
    * Otherwise $2n_1 - 1 + \delta < \lceil \frac{N-1}{g} \rceil$, the schedule length is determined by the number of nodes and g. As long as g sink children have not received the slots they need, MODESA maintains busy the g sink interfaces. Furthermore, MODESA uses a variable Round Robin that ensures that the schedule of the

nchild modulo g last sink children is never entirely postponed at the end. This is the difference with FlipFlop, where more than g uncomplete slots (i.e. slots with some sink interfaces inactive) may exist.

In both cases, MODESA maintains busy the g interfaces of the sink as long as possible. It is not possible to use a lower number of slots to schedule the sink children. Hence MODESA is optimal for multilines.

- Third case: balanced trees
  - When there is a single group, all subtrees are scheduled in parallel. The number of slots is the one needed to schedule the first subtree. Hence, MODESA is optimal.
  - When there are two groups, MODESA schedules in the first slot the sink children of the first group because of their higher number of packets to receive and in the second slot the sink children of the second group. Notice that depending on the number of sink children in the second group, some of the g interfaces of the sink may be unused. In the third slot, it is again the sink children of the first group. The sink children of the second group occupy the fourth slot. Hence, MODESA schedules in round robin the sink children of the first group and the sink children of the second group. Other nodes are scheduled in the slots where they do not conflict. MODESA uses the same number of slots as the FlipFlop algorithm. It is optimal.
  - When there are more than two groups, MODESA schedules successively in the $\lfloor \frac{nchild}{g} \rfloor$ first slots the sink children of the $\lfloor \frac{nchild}{g} \rfloor$ first groups. We distinguish two subcases:
    - If the last group contains exactly g sink children, MODESA repeats the $\frac{nchild}{g}$ first slots to schedule the sink children until they have transmitted all their messages.
    - Otherwise the behavior of MODESA differs a little insofar as in the $\lceil \frac{nchild}{g} \rceil^{th}$ slot, it schedules the $p < g$ sink children of the last group with the $g - p$ first sink children of the first group. It continues with in the next slot, the p last sink children of the first group with the $g - p$ first sink children of the second group...

  In both cases, MODESA maintains busy the g interfaces of the sink as long as possible. It is not possible to use a lower number of slots to schedule the sink children: MODESA is optimal for balanced trees.

□

Notice that in the example given in the proof of Theorem 5 in the previous chapter, MODESA provides 8 slots, the optimal number while FlipFlop is not optimal.

5.5 Performance evaluation of MODESA

We developped a GNU Octave [Octave] based simulation tool and performed simulation with different WSN topologies. We suppose that all nodes except the sink have a single radio interface and we vary the number of sink radio interfaces from 1 to 3. We make vary N the number of nodes from 10 to 100 and generate random trees where the maximum number of children is limited to 3. We use a Galton-Watson process as a branching stochastic process to generate random trees: each node gives birth to a random number of children independently of the others and according to the same distribution. In addition, we assume that the only topology links are those in the tree. Unless otherwise stated, all the following investigations adopt these values. In the following, each result is an average of 20 runs for small topologies ($\leq$ 30 nodes) and 100 runs for large topologies.

5.5.1 MODESA with homogeneous traffic

5.5.1.1 Evaluation of the optimality of MODESA for homogeneous traffic

We first evaluate the optimality of MODESA considering two types of configurations, depending on whose factor imposes the optimal schedule length:

- The size of the most populated subtree, denoted type T_t, where $2n_1 - 1 + \delta > \lceil \frac{N-1}{g} \rceil$;
- The number of nodes and g, denoted type T_n, where $2n_1 - 1 + \delta \leq \lceil \frac{N-1}{g} \rceil$.

As depicted in Figure 7.24, we notice that in random trees with 100 nodes, MODESA is optimal in respectively 89% of the T_t configurations tested and 74% of the T_n configurations tested, respectively. This illustrates the merit of MODESA.

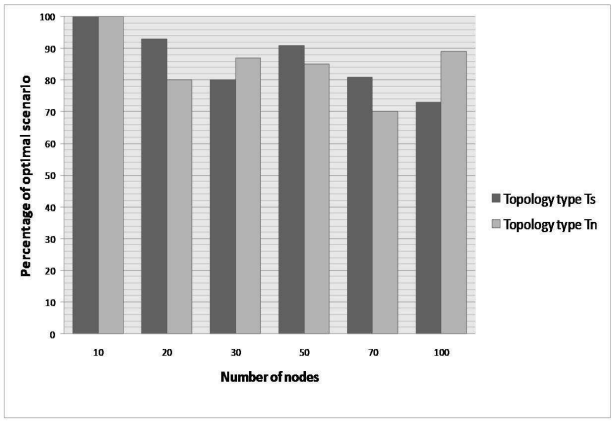
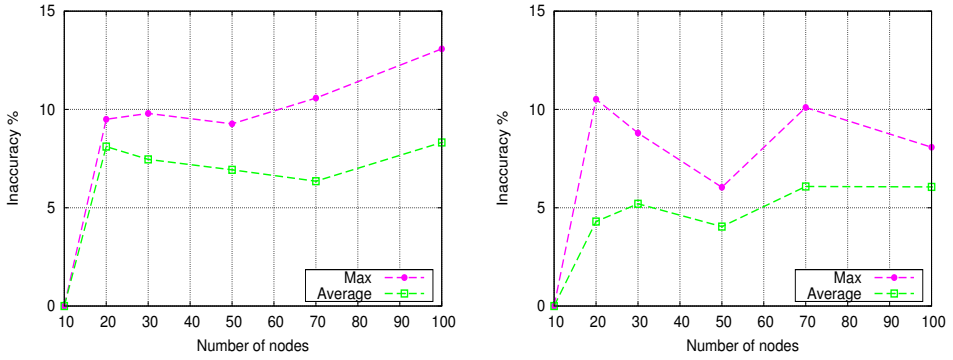


Figure 5.2: Optimality of MODESA in T_t and T_n configurations.

We now consider only the configurations where MODESA is not optimal and quantify the drift between MODESA and an optimal schedule with regard to the number of slots needed. For this purpose, we evaluate for each type of configuration the schedule length obtained by MODESA, denoted L_{MODESA} and the optimal one, $L_{Optimal}$. We compute the inaccuracy of

MODESA as $\frac{L_{MODESA}-L_{Optimal}}{L_{Optimal}}$. The inaccuracy of MODESA is depicted in Figures 5.3(a) and 5.3(b) for topologies of type T_t and T_n respectively. The maximal inaccuracy is 13% for the T_t and 10.5% for the T_n configurations, demonstrating the very good behavior of MODESA. The average inaccuracy in both T_t and T_n configurations is below 8.5%.



(a) (b)

Figure 5.3: Inaccuracy of MODESA in (a) T_t configurations (b) T_n configurations.

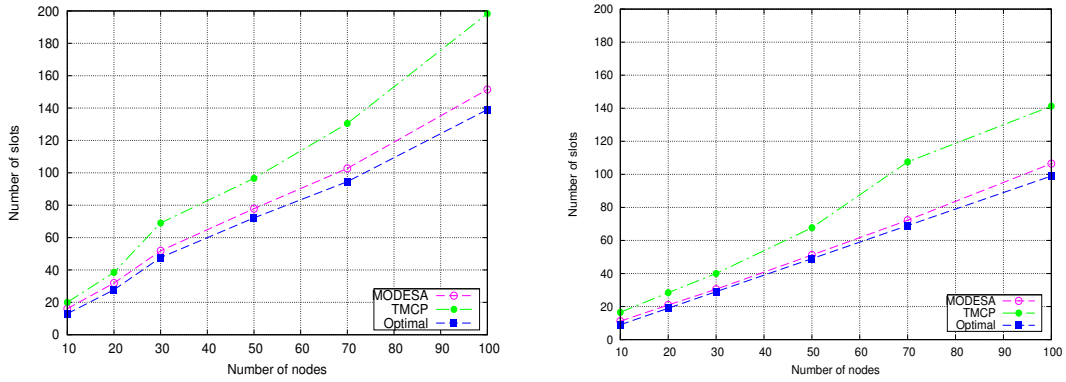
5.5.1.2 Comparison with TMCP

We compare MODESA to TMCP, a relevant cluster-based multichannel scheduling. We recall that TMCP partitions the tree topology into multiple subtrees. The inter-tree interference is minimized by assigning different channels to subtrees. All nodes in the same subtree communicate on the same channel. For TMCP, we set the priority of a node equal to its depth. Here, we compare MODESA and TMCP and differentiate two cases: homogeneous and heterogeneous traffic.

In the first set of simulations, all nodes generate one packet and the sink is equipped with a single radio interface. The number of channels is equal to the number of subtrees. We compare the number of slots needed to complete the convergecast, number of buffers, the number of radio switches (from active to sleep states and from sleep to active states) and the slot reuse ratio. Results are presented in Figures 5.4 and 5.5. Overall, the performance of MODESA is better than TMCP. Taking a closer look into the results plotted in Figure 5.4, we find that in configurations with 100 nodes, MODESA uses respectively 20% less slots in T_t configurations (23% in T_n configurations) than TMCP.

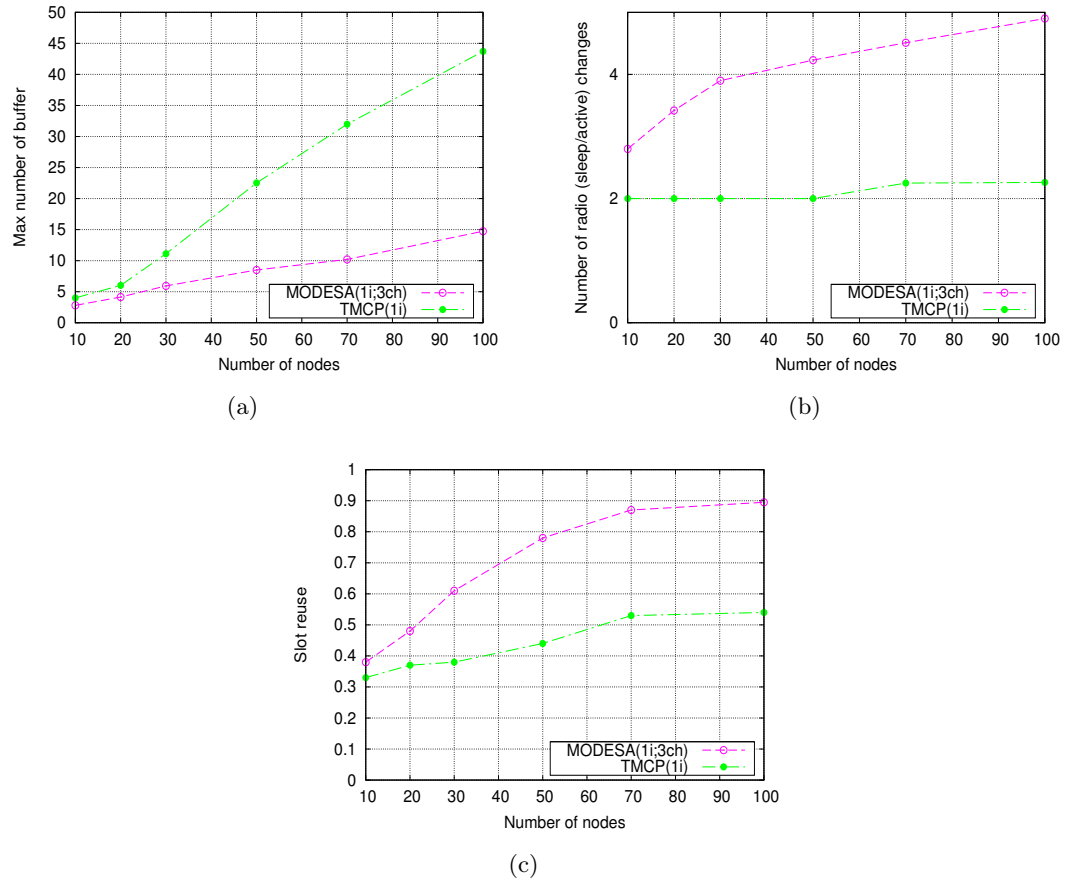
This agrees with evaluation results in Figure 5.5(c), where our joint channel and time slot assignment achieves more slot reuse ratio than TMCP. Moreover, the drift of MODESA from the optimal is still within 9% in T_s configuration (respectively 7% in T_n configuration) as depicted in the Figure 5.4. Furthermore, note that TMCP requires more buffers than MODESA: in 100 nodes topologies, MODESA needs only 15 buffers while TMCP requires 44 buffers as illustrated in Figure 5.5(a). This can be explained by the fact that MODESA takes into account the number of packets in the buffers when it schedules nodes.

Figure 5.5(b) plots the average number of radio (sleep/active) changes as a function of



(a) (b)

Figure 5.4: Scheduling with multiple channels: performance of MODESA and TMCP regarding the number of slots in (a) T_t configurations (b) T_n configurations.



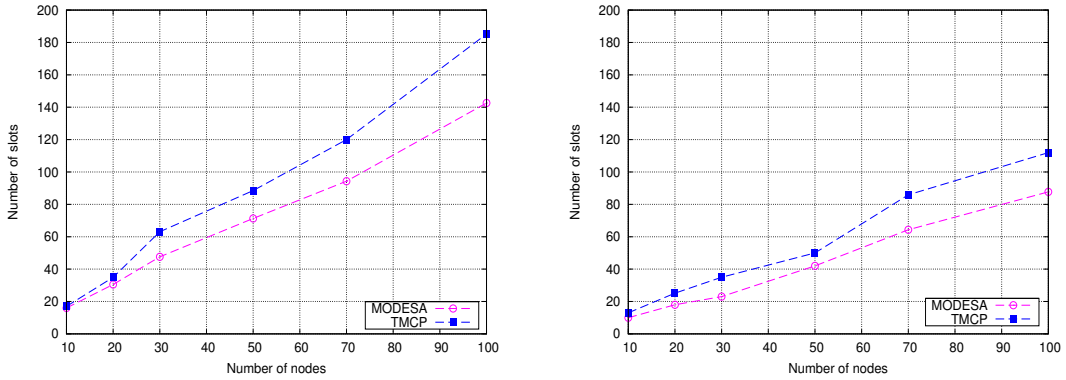
(a) (b)

(c)

Figure 5.5: Scheduling with multiple channels: performance of MODESA and TMCP regarding (a) max buffer (b) switch state (c) Slot reuse ratio.

number of nodes. MODESA is more greedy that TMCP. Note that for TMCP this metric is almost flat. This means that increasing the number of nodes does not yield more radio switches.

To further study the behavior of MODESA and TMCP, we conducted simulations where the sink is equipped with a number of radio interfaces equal to the number of subtrees. Each radio interface operates on a different channel. So the sink can receive simultaneously from its children.



(a) (b)

Figure 5.6: Scheduling with multiple channels and multiple radio interfaces for sink in (a) T_t configurations (b) T_n configurations.

Figure 5.6 shows the same behavior of curves as seen in Figure 5.4. Indeed, for small topologies (≤ 30), MODESA and TMCP are close. But when the number of nodes increases the gap between the compared algorithms is huge. These results unambiguously display MODESA's excellent performances in schedule length.

5.5.1.3 Evaluation of other metrics

We now focus on our joint time slot and channel assignment. We detail the performances of MODESA in terms of schedule length, throughput, radio switches per node in a schedule and buffer size, considering various number of radio interfaces.

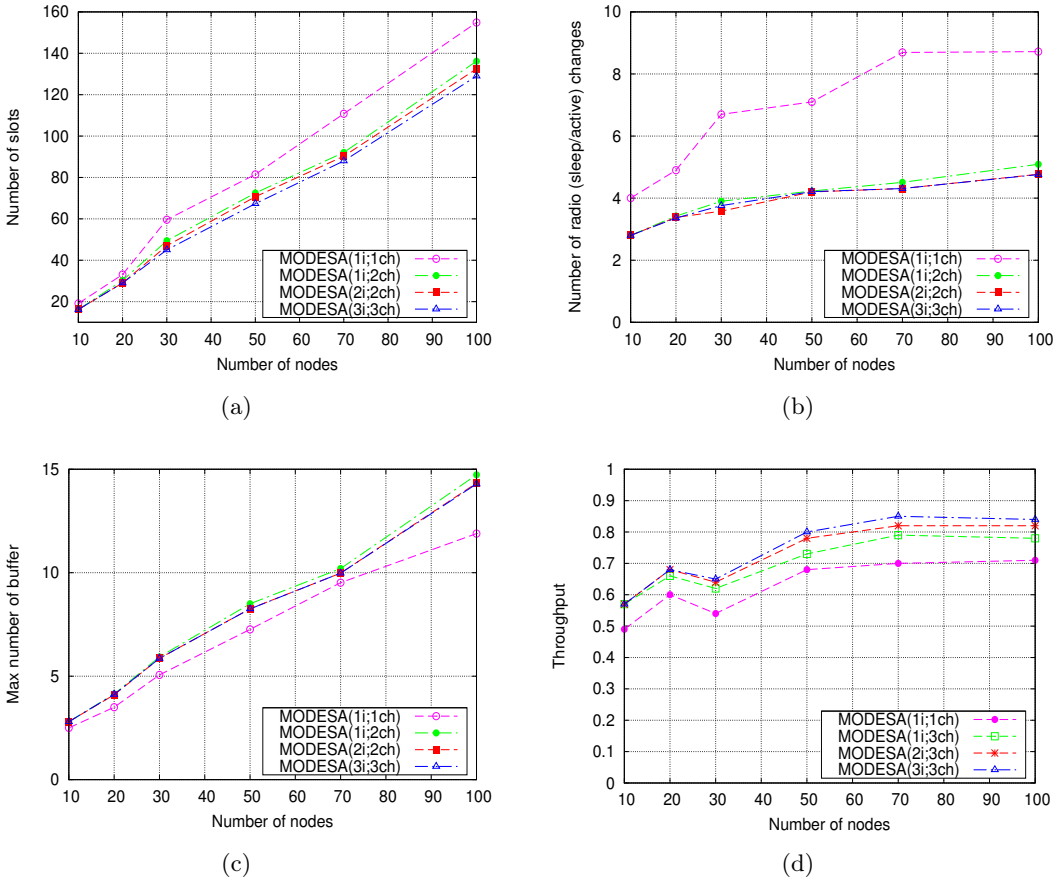


Figure 5.7: MODESA performance regarding (a) the number of required slots (b) the number of radio switches (c) the maximum buffer size (d) the throughput.

Figure 5.7(a) depicts the total number of slots for MODESA considering different numbers of channels. We observe that even when the sink has a single radio interface, the use of multichannel drastically decreases the TDMA cycle length: for example with 100 nodes, the use of only two channels decreases the number of slots by 12.82% (20 slots). With a single interface of the sink, the best performances are achieved when the number of channels is equal to two. When the sink is equipped with multiple interfaces, we observe also a reduction of the number of slots. Moreover, we notice that there is no interest to equip the sink with a number of radio interfaces greater than the number of its children. We also observe that it is useless to have a number of channels higher than the number of sink interfaces when this latter is greater than 1.

In addition, reducing the radio state switches is crucial to save the energy of sensors.

Therefore, for each node, we compute its number of switches as the number of times the node alternates between the sleep and active radio states in a cycle. Figure 5.7(b) shows that the number of switching between active ad sleep states is decreased by the only use of two channels for a sink with one or two interfaces, and a number of channels equal to the number of sink interfaces otherwise. Another parameter which directly impacts the execution of MODESA is the maximum number of buffers. So we also evaluate the maximum number of buffers required in a node during a TDMA cycle. Figure 5.7(c) demonstrates that MODESA with a single channel ensures the smallest number of required buffers. In multichannel wireless networks, the single radio interface uses more buffers than multi radio interfaces. This can be explained by the parallel transmissions allowed by the presence of at least two channels.

Finally, we evaluate the throughput. This latter is defined as the number of times where the sink receives packets divided by the total number of slots. As illustrated in Figure 5.7(d), the use of multi radio interfaces achieves higher throughput. Moreover, in case of sink equipped with one radio interface, the use of multichannel guarantees higher throughput than the single channel network.

5.5.2 MODESA with heterogeneous traffic

It is worth noting that in a WSN, we can have different types of wireless sensors running at possibly different sampling rates. This results in non-homogeneous initial demand of nodes.

5.5.2.1 Evaluation of the optimality of MODESA

To investigate how heterogeneous initial traffic can influence the optimal values and the behavior of MODESA and TMCP, we run these solutions on the examples depicted in Figure 5.8.

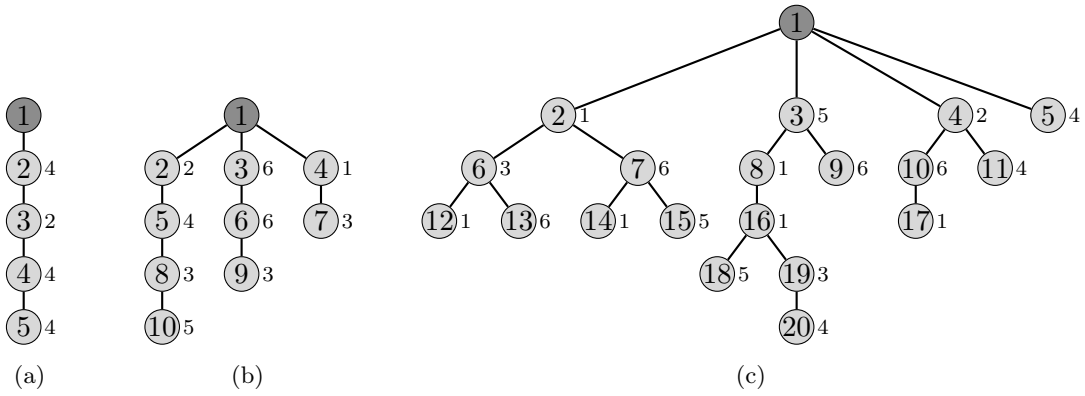


Figure 5.8: The optimal number of slots, *nbOptimalSlots*, for various topologies with different numbers of sink interfaces, $ninterf$, and channels, $n_{channel}$, with the notation: ($ninterf$;$nchannel$)=*nbOptimalSlots*. (**a**) (1;1) = 24; (**b**) (3;3) = 26; (**c**) (4;4) = 45.

For all these topologies MODESA is optimal. However, TMCP needs respectively 32, 34 and 58 slots in the line (a), multiline (b) and tree topologies (c). In addition, for the

multiline and tree topologies, MODESA needs only two radio interfaces and two channels to reach the optimal number of slots to complete convergecast. However, TMCP, even when the sink is equipped with 3 or 4 radio interfaces, does not achieve the optimal values. This can be explained by the fact that scheduling all the subtree on a single channel cannot ensure high spatial reuse ratio.

We have also compared the number of slots obtained by MODESA with the optimal number of slots in topologies with 100 nodes. For a sink with one radio interface and three channels, MODESA is optimal in 56% of the T_t configurations and in 85% of the T_n configurations as illustrated in Figure 5.9.

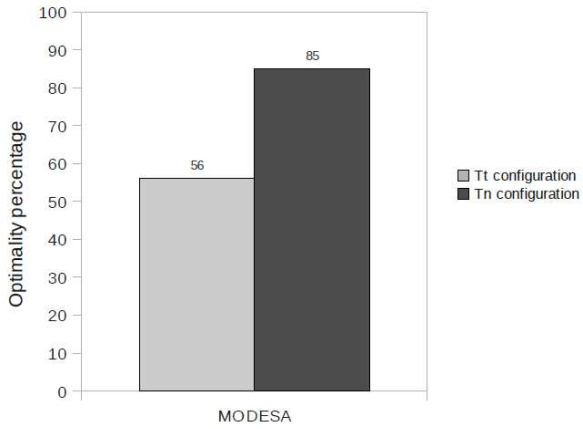


Figure 5.9: Optimality of MODESA rin T_s and T_n configurations with heterogeneous traffic.

In addition, we evaluate the drift of MODESA to the optimal schedule in configurations where they are not optimal. This distance is called inaccuracy and is computed as: $\frac{number\ of\ slots\ needed - optimal\ number\ of\ slots}{optimal\ number\ of\ slots}$. As depicted in Figure 5.10, the average inaccuracy of MODESA is 8.5.2% for the T_t configurations and 6.2% for the T_n configurations confirming the results with homogeneous traffic.

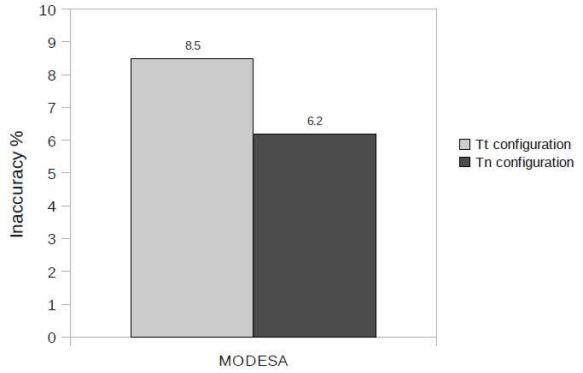


Figure 5.10: Inaccuracy of MODESA in T_s and T_n configurations with heterogeneous traffic.

5.5.2.2 Comparison with TMCP

In the following, we conduct simulations in larger number of configurations. The sink is equipped by a number of radio interfaces equal to the number of subtrees. We focus again on the number of slots needed to complete convergecast with heterogeneous demands.

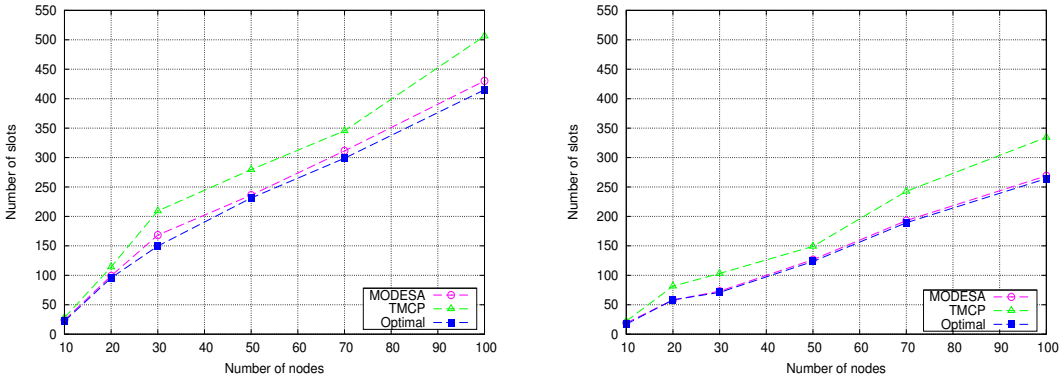


Figure 5.11: Scheduling with multiple channels and multiple radio interfaces for sink in (a) T_t configurations (b) T_n configurations.

The results shows again that MODESA is close to the optimal values of slots number: the distance is 5% in T_t configurations (respectively 3% in T_n configurations). In addition, MODESA obviously outperforms TMCP.

To summarize, the results obtained show that MODESA, our proposed scheduling technique, is capable of providing schedules that are at most 10% far from the optimal by intelligently assigning slots and channels to nodes based on a dynamic priority.

5.5.3 Impact of additional links

In previous simulations, we focused on computing spatial-reuse TDMA schedules once interfering links, that do not belong to the routing tree, have been eliminated. This is a crucial assumption for the work of Incel et al. [Incel 2012]: they proposed JFTSS, a scheduling approach for data convergecast. JFTSS is applied after a receiver-based channel allocation step that eliminates most of the interferences. Hence, the convergecast schedule depends on the efficiency of the channel assignment solution. Nevertheless, it was proven that [Ghosh 2009] assigning a minimum of channels to receivers such that all interfering links are removed, is NP-complete. It is important to note that our proposed algorithm MODESA does not require that all interfering links are removed. It easy takes into account the presence of additional interfering links. Indeed, it is a black box for the algorithm.

We run MODESA through an example shown in Figure 5.12 to explain how using multiple channels can reduce the impact of interfering links on schedule length. The solid lines denote the tree links while dashed lines denote the additional interfering links. These additional links have been added to Figure 5.12(b) and Figure 5.12(c).

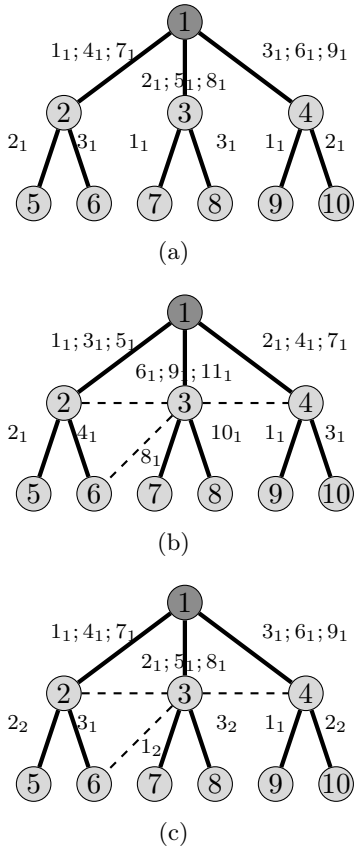


Figure 5.12: Convergecast scheduling provided by MODESA with the notation $Slot_{channel}$: (a) Schedule length of 9 slots using single channel (b) Schedule length of 11 slots when interfering links are added (c) Schedule length of 9 slots when interfering links are added and 2 channels are used.

The first subfigure 5.12(a) indicates that 9 slots are needed to complete convergecast. However, when additional interfering links are added, 2 extra slots are required (see Figure 5.12(b)). As illustrated in last subfigure 5.12(c), adding an additional channel for scheduling re-establishes the initial schedule length because more parallel transmissions are allowed.

To further investigate the impact of interfering links on schedule length, another set of simulations is conducted. In the results presented after, the sink is equipped with a number of radio interfaces equal to the number of children. the number of channels is equal to the number of radio interfaces. Additional links are added: for each node at even depth d in the tree, an additional link is generated with a node at depth $d-1$ different from its parent. Furthermore, with a probability equal to 0.5, another link is added with a node of depth $d+1$ different from its children. In average, 60% additional links are added.

As it can be seen in Figure 5.13, the impact of additional links depends on the routing tree. The worst routing tree is T_s for both MODESA and TMCP. For 100 nodes, MODESA needs 13 additional slots to complete convergecast in T_t configurations while only 5 slots are needed in T_n configurations. It is also worth noting that, for MODESA, the additional

number of slots due to additional links is smaller than this for TMCP. This shows the capacity of MODESA to incorporate easily the additional conflicting links.

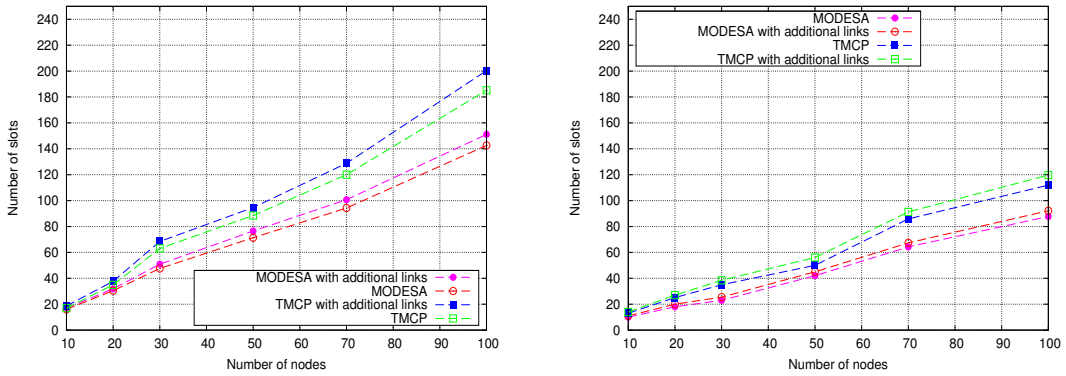


Figure 5.13: Impact of additional interfering links on scheduling in (a) T_t configurations (b) T_n configurations.

To summarize this section, MODESA relies on an efficient heuristic that attains a schedule length close to optimal values. It is significantly better than the state-of-the art TMCP solution. The gain can reach up to 20%. It also incurs less buffer consumption. Another advantage of MODESA relies on its flexibility to take into account additional interfering links.

5.6 MODESA improvement

In this section, we seek to further enhance the basic version of MODESA by adding other options or extensions. These latter will tackle the problem of channel load balancing, multipath routing and finally topology diversity on different channels.

5.6.1 Channel allocation strategy

Basic MODESA has the goal of achieving conflict-free schedules so that data convergecast latency is minimized. MODESA has been presented with the greedy variant for channel allocation. In this variant, channels are allocated in increasing order that is the same for all time slots. Nevertheless, the greedy channel allocation strategy does not ensure channel load balancing. That is why we focus on different variants of MODESA that tend to balance the channel load.

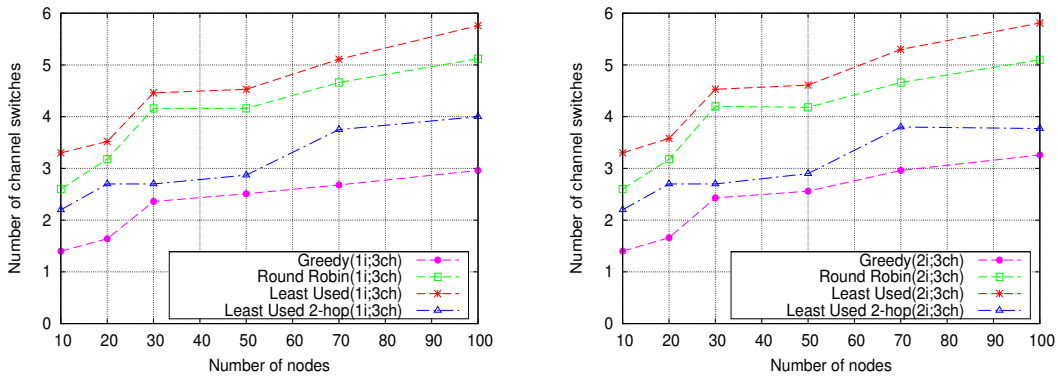
1. *Round Robin:* channels are considered in a circular order, depending on the current time slot.

2. *Least used channel:* we first favor the least used channel among the whole network (i.e. the channel with the smallest number of transmissions).

3. *Least used 2-hop channel:* on any node up to 2-hop from the selected node, we compute the maximum or average load of any channel and select the channel with the least load.

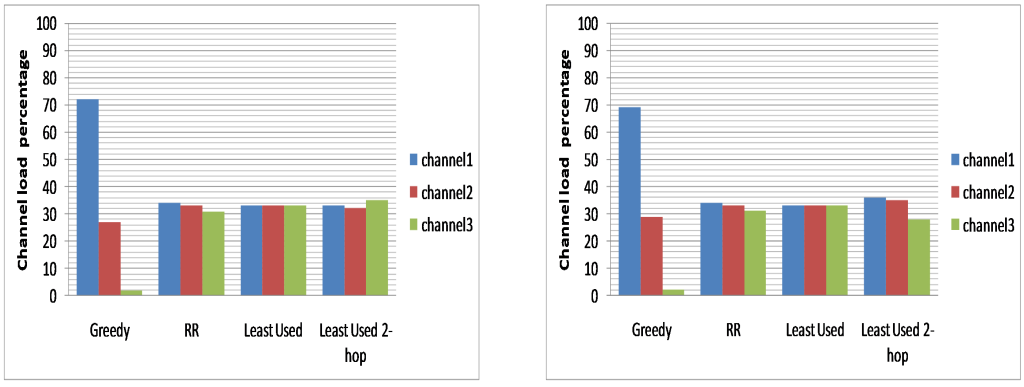
We notice that all these variants of MODESA provide the same number of slots.

On the first hand, we analyze the impact of variants of MODESA on the number of channel switches per node. Figure 5.14 shows that all these variants of MODESA achieve a small number of channel switches leading to a minimized medium access time. As expected, the greedy variant achieves the lowest number of switches. As illustrated in the Figures 5.14(a) and 5.14(b), greedy has the same behavior in the two curves. This can be explained by the fact that with one or two interfaces for the sink, two channels are sufficient to schedule nodes transmissions. The greedy variant does not use the third channel.



(a) (b)

Figure 5.14: Channel switching in case of sink equipped with (a) 1 interface and 3 channels (b) 2 interfaces and 3 channels



(a) (b)

Figure 5.15: Channel load in topology with 100 nodes and sink equipped with a) 1 interface and 3 channels (b) 2 interfaces and 3 channels

Figure 5.15 depicts channels loads achieved by different MODESA variants. As illustrated, the least used variant outperforms greedy, Round Robin and least used 2-hop in balancing the number of times a particular channel is used. Hence, it minimizes the co-channel interferences. However, Round Robin provides the best trade off between the implementation simplicity and a channel load balancing.

Notice also that when channels have different qualities, it is better to use the greedy variant where the first channel considered by MODESA is the channel with the best quality, the second channel considered has the second best quality and so on. As a consequence, the channels with the best qualities are more used than the channels with bad quality, resulting in more robust communication.

5.6.2 Multipath transmission scheduling

We investigate here multipath routing extension for MODESA, i.e allowing a traffic flow to be split among several paths in order to balance link utilization and minimize schedule length.

5.6.2.1 Motivations

Although route discovery through single path is less expensive than multipath approach, the limited robustness of a single path against node/link failure, reduces the reliability and throughput required by some convergecast applications. Indeed, the dense deployment of sensor nodes makes easy the existence of several paths from any sensor node towards the sink. Furthermore, multipath approach provides more opportunities for a node to transmit due to the existence of multiple parents. Indeed, it involves multiple forwarding candidates at each hop and the actual forwarder is selected according to some metrics. Thus, multipath approach contributes to reduce the end-to-end delays for data convergecast. In addition, available paths can be used to balance network load among nodes specifically in data intensive applications. Hence, it limits the exhaustion of the scare energy budget of sensors.

This section investigates the merging between multichannel communications that allow more parallel transmissions and multipath routing. This latter finds multiple paths. Hence, each node has multiple candidate parents. Then, the scheduler orchestrates the nodes activities selecting a parent for each transmission and assigning a time slot and channel. The goal is to ensure a schedule with the lowest possible latency.

That's why we focus in this section on taking advantage of multipath routing to provide a schedule with minimized latency in multichannel WSNs. We first give a brief summary of works that tackles this problem in multichannel WSNs. Subsequently, we present a version of MODESA that supports multipath routing in order to capture the gains of concurrent transmissions of multipath.

In [Li 2011b], authors tackle the problem of joint routing, scheduling and power transmission assignment. Since varying transmission power results on higher interference level, multichannel multiradio paradigm is used to alleviate the impact of interferences. They propose an interference-degree-based greedy algorithm for computing Concurrent Transmissions Link Sets (CTLS). This solves the channel and scheduling problem. A random walk-based heuristic was proposed to solve the multipath routing problem.

Authors of [Zhang 2011] address the joint problem of routing and scheduling under a physical interference model. Their solution incorporates two components: (1) Power control algorithm that allocates optimal transmission range and transmission rate for each node. (2) joint routing and scheduling that assigns for each link, a couple (channel, slot) and searches for multiple paths for source node.

The above mentioned studies do not deal with the convergecast applications and consider transmission power adjustment. As demonstrated in [Incel 2012], multichannel paradigm is more efficient than transmission power adjustment for mitigating interferences.

5.6.2.2 The extended version of MODESA supporting multipath routing

We extend the single path routing in MODESA to support multipath routing as follows:
(1) a node has a set of potential parents that can be candidate receivers of data packets. For a node having depth d, any $1-hop$ neighbor node, whose depth is equal to $d-1$, is a potential parent. Indeed, as data packets should be forwarded towards the sink, the potential parent should be a $1-hop$ neighbor with a lower depth in the routing tree. We denote by $NbParent(u)$, the number of parents of node u.
(2) The $Rcv(Parent(u))$ takes into account all the packets that $Parent(u)$ should receive from its children in the different paths.
(3) the priority of a node is modified to take into account the presence of multiple candidate parents. The priority of a node u is equal to $Prio(u) = \frac{\sum_{i \in Parent(u)} Rcv(i) * RemPckt(u)}{2^{NbParent(u)-1}}$. We recall that $RemPckt(u)$ is the number of packets the node has in buffer at the current iteration
(4) MODESA will search the t-uple(u, $p = Parent(u)$, channel, slot) such this transmission does not interfere with already scheduled transmissions on the selected time slot and channel.
(5) Let us denote by $p = Parent(u)$, the actual receiver of the packet of node u. We update the value of Rcv for any other candidate parent of u that have not been selected for the transmission. Indeed, for each $q \in Parent(u) \neq p$, $Rcv(q) = Rcv(q) - 1$.

Figure 5.16 gives an example of the scheduling obtained by multipath routing with MODESA. Two channels are available at each node and the sink is equipped by two radio interfaces. Immediate acknowledgment are not taken into account. As illustrated by subfigure 5.16(a), MODESA without multipath routing needs 7 slots to complete convergecast. This is two slots less than without the existence of multiple parents (see subfigure 5.16(b)).

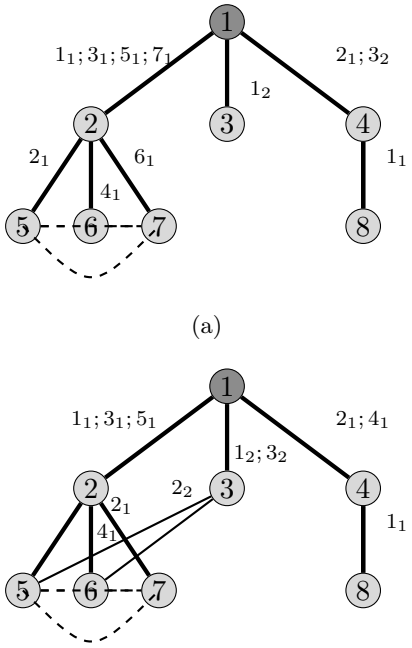


Figure 5.16: Illustrative example of multipah routing.

To highlight the performance, we carried out two experiments. The evaluation focuses on two metrics: latency (number of slots needed for convergecast) and the average maximum buffer. Furthermore, MODESA multipath is compared to the basic version of MODESA. Figure 5.17 evaluates the number of slots needed by MODESA multipath in comparison with MODESA basic. Generally, in both configurations (T_t and T_n), MODESA with multipah routing performs better and ensures smaller latency. With multipath routing, schedule length is reduced by 10% in T_s (respectively 8.8% in T_n). This can be explained as follows: in MODESA basic version, a node has only one parent to send its data. If this parent is occupied by the reception from another child, this node should wait the next time slot to transmit. Nevertheless, in MODESA with multipath routing, a node has a set of candidate receivers. Hence, if one parent is not available, it can check the other parents for possible transmission.

Furthermore, we also explore the trade off between multipath routing and maximum number of buffer recorded at a node.

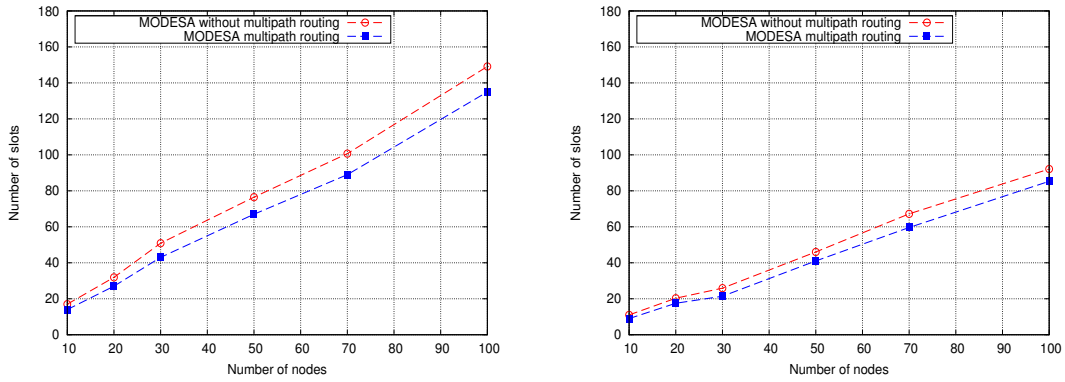


Figure 5.17: Impact of multipath routing on Schedule length in (a) T_t configurations (b) T_n configurations

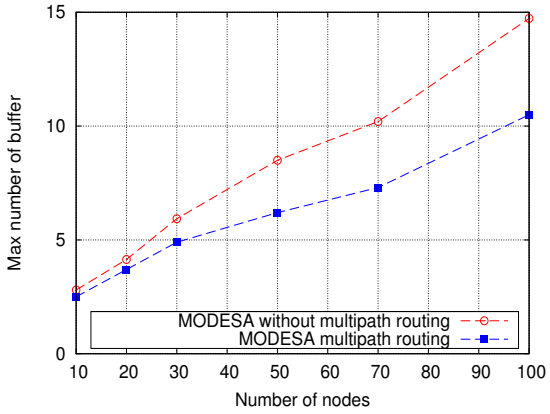


Figure 5.18: Impact of multipath routing on buffer size.

Compared with basic MODESA, MODESA with multipath routing consumes less buffers as we can see from Figure 5.18. The buffer consumption of MODESA without multipath routing increases faster with the number of nodes. For example, for 100 nodes, MODESA with multipath routing needs on average 2/3 the buffer size of MODESA basic version.

It appears that multipath routing is an elegant way to provide shorter schedule length. In addition, multipath routing ensures fault tolerance. Indeed, sensor nodes benefit from the availability of alternative paths to transmit their data packets despite a parent failure. Hence, data convergecast can be continued without interruption. Moreover, multipath routing ensure load balancing by splitting traffic over several paths.

5.6.3 Different topologies on different channels

As mentioned in chapter 3, a multichannel WSN may have different topologies as the wireless nodes operate on different channels. The topology can be connected on all channels whereas

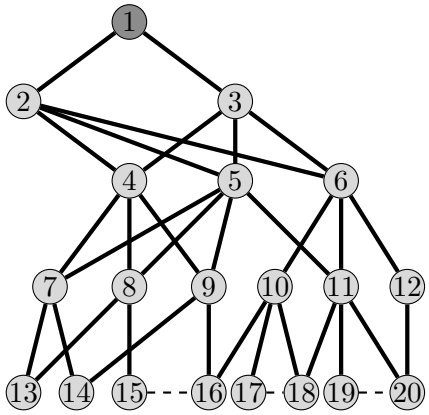
the topology may differ from one channel to another. Hence, the neighborhood set of a node can differ from one channel to another. Thus, the conflicting set of a node depends also on the channel selected for transmission.

Our solution MODESA makes no assumption about the neighborhood of a node on a channel c and the neighborhood of the same node on a channel c'. Nevertheless, it is probable that the differences in neighborhood will be small if 802.15.4 channels are not disturbed by other RF sources. However, if the full connectivity is not achieved on a given channel, MODESA will only use the connected component including the sink. This allows the sink to collect the topology and traffic information needed as input by MODESA.

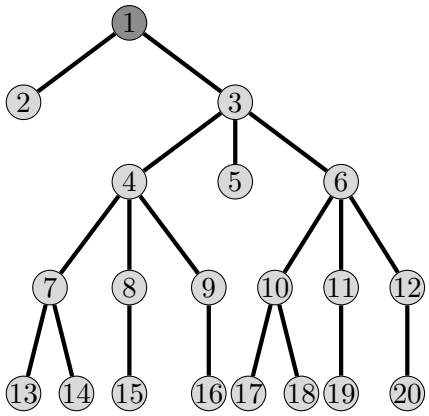
We make here some observations how MODESA can adapt to links diversity from one channel to another. Some variables are redefined to accommodate to topologies diversity on channels. For a node u, we redefine:

- $Depth(u) = min_{c \in channels}\ Depth(u, c)$
- $Parent(u, c)$=set of nodes that are eligible to be a parent of u on channel c ($Depth(Parent(u, c)) < Depth(u, c)$).
- $BlackListChannel(u)$=set of channels such that $Depth(u, c) > Depth(u)$
- $Conflict(u, c)$= set of conflicting nodes with u on channel c
- $Parent(u) = \cup Parent(u, c)$, with $c \notin BlackListChannel(u)$

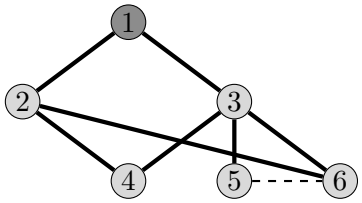
Varying the logical topology from one channel to another, we are interested in the following issue: How the different logical topologies can alter the scheduling and namely the schedule length. To get initial answers to this question, we run MODESA on typical examples illustrating topology diversity from one channel to another. Figure 5.19 depicts a typical example. The channel $c1$ is the most constraining channel because it includes a higher number of nodes and links than other channels. The less constraining channel is channel $c3$: only nodes from depth 1 and 2 exist on this channel. To investigate how the topology diversity affects the scheduling, we conduct two simulations: in the first (respectively the second) simulation, channels are assessed in the decreasing (respectively in the increasing) degree of conflict constraints. The sink is equipped with a single radio interfaces and 3 channels are available.



(a) logical topology on channel 1



(b) logical topology on channel 2



(c) logical topology on channel 3

Figure 5.19: Topology diversity on different channels.

The schedule length obtained in the first simulation is 22 slots while it is equal to 20 slots in the second simulation. This observation confirm the results obtained from other tested examples. MODESA consistently yields a better performance when the channels are selected in the increasing degree of conflicting constraints.

5.7 Conclusion

Applying WSNs in industrial environment requires fast and reliable data convergecast. In this chapter, we have shown how multichannel communications contribute to achieve these requirements. We focused on the problem of finding conflict-free schedules that minimizes the data convergecast delay. Our key results in this chapter are twofold. First, we have proposed an optimal integer programming-based solution that assigns jointly time slots and channels to nodes. This model was implemented using the GLPK tool to find optimal schedules to complete convergecast. Because, these solutions have an exponential running time as the network size and nodes traffic increase, scheduling has to rely on heuristics. Thus, as a second contribution, we have proposed the MODESA algorithm and have proved its optimality in many multichannel topologies of WSNs. Simulation results show that MODESA outperforms significantly TMCP, a relevant work on data convergecast in multichannel WSNs. MODESA needs a small buffer size and reduces the number of radio active/sleep switches per node in a cycle. In addition, we described variants of MODESA that balance traffic load between channels. Furthermore, we can improve MODESA by adopting the behavior of FlipFlop when there are exactly one or two groups, making it optimal even in case of unbalanced trees. Another big advantage of MODESA is its conceptual flexibility. Unlike JFTSS, MODESA provides conflict-free schedules even when interfering links remain in the routing tree.

According to Table 3.1 in Chapter 3, MODESA can be classified in the semi-dynamic category that supports 2-level architecture. Indeed, MODESA can be applied for multiple data gathering cycles while there is no changes in nodes traffic demands or available channels list. Jointly with MAC layer, MODESA reduces delays and ensures energy efficiency.

In this chapter, we focused on data convergecast from sensors to a predefined sink. However, recently, there has been a renewal of interest in using multiple sinks for WSNs to achieve power saving and support many applications. The next chapter will tackle the problem of providing conflict-free schedules that minimize gathering delays in multi-sink WSNs.

Chapter 6

MUSIKA: a MUlti-SInk Slot Assignment for Convergecast in Multichannel WSNs

Contents

6.1 Introduction . . . 93
6.2 State of the art . . . 94
6.3 Network model and problem formalization . . . 95
6.3.1 Network model . . . 96
6.3.2 Multi-sink multichannel convergecast problem formulation . . . 96
6.4 MUSIKA: MUlti-SInK slot Assignment . . . 98
6.4.1 Principles . . . 98
6.4.2 Algorithm . . . 99
6.5 Illustrative example . . . 101
6.6 Performance evaluation . . . 102
6.7 Conclusion . . . 104

6.1 Introduction

The previous chapter dealt with the problem of minimizing data gathering delays. These data are collected from sensor nodes towards one sink in a multi-hop convergecast structure. However, some applications require many sinks to collect data. Indeed, the use of multiple sinks ensures:

1. A more reliable data gathering. This property is expected especially for critical information (path diversity), since in many deployments the channel used by the WSN may encounter perturbations.

2. Energy efficiency by decreasing both the load and the energy consumption of nodes close to the sink. Indeed, these nodes must forward a higher traffic toward the sink, which is a severe threat to network lifetime. The existence of several sinks enables a better load balancing between nodes.

3. Sinks running different functionalities of the application considered. This allows a higher flexibility in the mapping of application functionalities on wireless nodes. Indeed, this mapping may depend on several factors such as node location, desired redundancy degree and may take into account heterogeneous application requirements with regard to expected functionalities.

In this chapter, unlike studies that focus only on the time slot assignment problem, i.e. how to send data collected by multiple sources to a common sink and if possible in a minimum of time, we focus on a multi-sink multichannel context with a dedicated traffic per sink. Mainly we formalize the multichannel multi-sink convergecast problem in WSNs aiming at minimizing convergecast latency. We then propose our algorithm MUSIKA. Finally, performances in terms of cycle length, delivery delay and buffer size of the MUSIKA algorithm are evaluated by simulation in Section 6.6.

6.2 State of the art

Authors of [Macedo 2009] presents LEMMA, a spatial-reuse TDMA based scheduling protocol. It aims at minimizing latency for convergecast application where traffic is expected to be sporadic. LEMMA is a distributed solution. Each node has to compute a transmission time slot that avoids the interference from its neighbors and cascades its packets to its parents in the most suitable time slot. Indeed, the time slots are granted in the deceasing depth order providing a cascading assignment that minimizes latency. Moreover, the decision of slot assignment is based on the link quality directly experienced by sensors. To support multi-sink aspect, LEMMA adopts the concept of interference-free schedule of overlapping convergecast trees. Hence, a node can be involved in several allocations that belongs to different convergecast trees. This work has inspired us to adopt the interference-free schedules of overlapping convergecast tree concept.

Sixia et al. [Chen 2009] tackle the problem of latency minimization of raw data collected from sensors toward multiple sinks. An approximation algorithm is proposed to minimize the latency of data collection schedule. Authors show that it gives a constant-factor performance guarantee under the unit disk model. A heuristic algorithm is also presented based on breadth first search to distribute the data flow of the network. Nevertheless, their solution does not differentiate traffic. Indeed, the data stored at each node can be sent to any sink in the network.

A centralized data collection method called DD (Drainage Divide) was proposed in [Hiromori 2012] that targets periodic data collection toward multiple sinks. DD builds a set of disjoint spanning trees rooted at the sinks. In each tree, the delay from each node to the corresponding sink is kept under a given deadline. However, there is no guarantee on data delivery since a CSMA/CA protocol is used.

In [Mottola 2011], the routing problem from many sources to multiple sinks was investigated. Authors derived an analytical model that computes a joint routing and scheduling solution. A distributed heuristic is proposed to optimize routing over sink-rooted trees. The heuristic includes a routing quality that takes into account the number of source-sink paths passing through the node as well as the number of sinks that it is sending data to. However, authors do not consider raw data convergecast. Data from many sources are merged and forwarded together which is not the scope of our work.

One limitation of the above proposed time slot assignments for convergecast is that they do not involve the multichannel paradigm. The throughput requirements of many applications of WSNs is difficult to meet with a single wireless channel as argued in previous chapters.

In [Tan 2008], authors focus on time slot assignment in a multi-sink single hop Ultra Wide Band (UWB) WSN which is formulated as a linear programming problem. They also implement a heuristic to improve both throughput and fairness. They show that it is deemed scalable with multiple sinks. A drawback of this work is that it is limited to single hop networks.

None of the above related works about time slot assignment for convergecast deals with traffic differentiation in multi-hop multichannel multi-sink WSNs. That is why, this chapter proposes on the one hand a linear programming formalization of the problem and on the other hand a deterministic contention-free based algorithm called MUSIKA for convergecast in multichannel multi-sink WSNs.

6.3 Network model and problem formalization

The problem consists in orchestrating in a conflict-free manner the activities of nodes involved in overlapping convergecast trees. Each sink node is assumed to be the root of a convergecast tree. Each tree has its own cascading assignment of time slots. The solution should ensure that several joint time slot and channel allocations do not interfere with each other on convergecast trees.

In this section, we present a formalization of the multi-sink slot assignment problem under the assumptions 1, 2, 3, 5 and 6 given in the Chapter 4. This model should ensure that no two conflicting nodes transmit simultaneously on the same channel. In addition, to reflect traffic differentiation, the following assumption is added.

Assumption 7. *Differentiated traffic: Each traffic generated by a source node is tagged with its destination sink and must be transmitted to this sink. With each traffic is associated its importance degree from the application point of view. A traffic class groups all traffic with the same importance degree and the same sink as final destination. Each node maintains a FIFO queue per traffic class. In this work, we consider the general case where two sinks may have traffic with the same importance degree or different importance degrees.*

One important point that requires mentioning here is that the assumption 5 can be relaxed by considering packet retransmission to recover from packet losses. In this case, the amount of traffic should take into account these packets retransmissions. These latter can be evaluated considering an estimated packet loss rate.

Similarly assumption 3 can be relaxed considering that the slot size allows the transmission of $p > 1$ packets. In such a case, the traffic demand should be mapped into a slot demand.

We are looking for a multichannel slot assignment of minimum length ensuring that there are no two conflicting nodes transmitting simultaneously on the same channel.

6.3.1 Network model

The network is modeled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices representing the nodes of the network and $\mathcal{E}$ is the set of edges representing the communication links between nodes.

Let $\mathcal{V}_s$ be the set of sinks, with $\mathcal{V}_s \subset \mathcal{V}$. For each node $v \in \mathcal{V}$ and $s \in \mathcal{V}_s$ with $v \neq s$, we define $p_{v,s}$ the number of packets that v generates at each cycle and has to transmit towards the sink s. Moreover, for any node $v \in \mathcal{V}$, let $Conflict(v)$ be the set of conflicting nodes that interfere with v when transmitting on the same channel. Let $\mathcal{E}^+(v)$ denote the set of links through which a node $v \in \mathcal{V}$ can transmit. Let $\mathcal{E}^-(v)$ be the set of links through which a node $v \in \mathcal{V}$ can receive.

Let $\mathcal{C}$ be the set of channels usable for any transmission. The contention-free cycle is composed of at most T_{max} slots, where T_{max} denotes the maximum length of the cycle.

6.3.2 Multi-sink multichannel convergecast problem formulation

The optimization problem consists of minimizing the number of slots needed to complete the convergecast. To formulate the problem as a linear program, we introduce new variables. We define $a_{e,v,s,c,t}$ the activity of a link $e \in \mathcal{E}$ transferring a packet originated from $v \in \mathcal{V}$ towards $s \in \mathcal{V}_s$ on the channel $c \in \mathcal{C}$ in the slot t, ie $a_{e,v,s,c,t} = 1$ if and only if there is a transmission of a packet originated from v to s on the link e on the channel c in the time slot t and $a_{e,v,s,c,t} = 0$ otherwise.

Furthermore, let u_t be the use of a slot t, in other words $u_t = 1$ means that there is at least one link activity on at least one channel in the slot t and $u_t = 0$ denotes an empty slot. The objective is to minimize the number of slots t used in the cycle:

$$min \sum_{t=1}^{T_{max}} u_t$$

This objective is subject to:

$$\begin{aligned} a_{e,v,s,c,t} &\leq u_t \\ &\forall e \in \mathcal{E}, \forall v \in \mathcal{V}, \forall s \in \mathcal{V}_s, \\ &\forall c \in \mathcal{C}, t \leq T_{max}, \end{aligned} \tag{6.1}$$

$$\sum_{o\in\mathcal{V}}\sum_{s\in\mathcal{V}_s} a_{e,o,s,c,t}+\sum_{o\in\mathcal{V}}\sum_{s\in\mathcal{V}_s} a_{e',o,s,c,t} \leq 1 \quad \forall v \in \mathcal{V}, \forall e \in \mathcal{E}^+(v), \forall w \in Conflict(v), \forall e' \in \mathcal{E}^+(w), \forall c \in \mathcal{C}, t \leq T_{max} \tag{6.2}$$

$$\sum_{e\in\mathcal{E}^+(v)}\sum_{c\in\mathcal{C}}\sum_{t=1}^{T_{max}} a_{e,v,s,c,t} = p_{v,s} \quad \forall s \in \mathcal{V}_s, \forall v \in \mathcal{V} \setminus \{s\} \tag{6.3}$$

$$\sum_{e\in\mathcal{E}^+(v)}\sum_{c\in\mathcal{C}}\sum_{t=1}^{T_{max}} a_{e,w,s,c,t} = \sum_{e\in\mathcal{E}^-(v)}\sum_{c\in\mathcal{C}}\sum_{t=1}^{T_{max}} a_{e,w,s,c,t} \quad \forall s \in \mathcal{V}_s, \forall v \in \mathcal{V} \setminus \{s\}, \forall w \in \mathcal{V} \setminus \{v\} \tag{6.4}$$

$$\sum_{e\in\mathcal{E}^-(s)}\sum_{v\in\mathcal{V}}\sum_{c\in\mathcal{C}}\sum_{t=1}^{T_{max}} a_{e,v,s,c,t} = \sum_{w\in\mathcal{V}\setminus\{s\}} p_{w,s} \quad \forall s \in \mathcal{V}_s \tag{6.5}$$

$$\sum_{e\in\mathcal{E}^+(v)}\sum_{c\in\mathcal{C}} a_{e,v,s,c,t} \leq p_{v,s} - \sum_{e\in\mathcal{E}^+(v)}\sum_{c\in\mathcal{C}}\sum_{t'=1}^{t} a_{e,v,s,c,t'} \quad \forall s \in \mathcal{V}_s, \forall v \in \mathcal{V} \setminus \{s\}, t \leq T_{max} \tag{6.6}$$

$$\sum_{e\in\mathcal{E}^+(v)}\sum_{c\in\mathcal{C}} a_{e,w!,s,c,t} \leq \sum_{e\in\mathcal{E}^-(v)}\sum_{c\in\mathcal{C}}\sum_{t'=1}^{t} a_{e,w,s,c,t'} - \sum_{e\in\mathcal{E}^+(v)}\sum_{c\in\mathcal{C}}\sum_{t'=1}^{t} a_{e,w,s,c,t'} \quad \forall s \in \mathcal{V}_s, \forall v \in \mathcal{V} \setminus \{s\}, \forall w \in \mathcal{V} \setminus \{v\}, t \leq T_{max} \tag{6.7}$$

Constraint 6.1 binds the use of a time slot to at least the activity of one link on any channel in this slot. Constraint 6.2 guarantees that two conflicting nodes v and w do not transmit on the same channel in the same time slot.

Constraint 6.3 ensures that for each sink s, any non-sink node v transmits during the cycle all the packets it has generated for s. Constraint 6.4 expresses that for each sink s, any intermediate node v forwards towards s all the received packets destinated to s. Constraint 6.5

ensures that each sink s receives all the packets generated in the WSN with final destination s.

Constraint 6.6 expresses the rule that a node v transmits a packet towards a sink s at the slot t if and only if its buffer of the traffic destinated to s is not empty. Constraint 6.7 guarantees that any sink s forwards the traffic addressed to another sink it receives.

6.4 MUSIKA: MUlti-SInK slot Assignment

In this section, we present MUSIKA, a centralized raw data convergecast scheduling algorithm for multichannel multi-sink WSNs. MUSIKA is based on our previous work MODESA. However, this latter does not take into account the existence of multiple sinks and traffic differentiation. Therefore, we propose a novel solution to address this new context.

6.4.1 Principles

MUSIKA proceeds slot by slot to build the multichannel multi-sink schedule, applying the following rules:

R1. Only nodes having at least one packet to transmit compete for the current time slot. They are ordered according to their decreasing priority. Let $\mathcal{N}$ be this ordered set.

R2. The competing node in $\mathcal{N}$ with the highest priority is selected first.

R3. A node allowed to transmit in the current slot will transmit the first packet in the FIFO queue of the traffic class with the highest importance degree. If several traffic classes have the same importance degree, the first packet of the longest queue in these traffic classes will be chosen.

R4. A node is allowed to transmit in the current slot if and only if:

1. this node and its parent in the data gathering tree corresponding to traffic class, have an available radio interface;
2. there exists a channel where this node does not conflict with nodes already scheduled in this slot.

R5. The next node to be selected is the next one in $\mathcal{N}$. It is allowed to transmit according to rule R3 and will transmit its packet selected according to rule R4. And so on until all nodes in $\mathcal{N}$ have been checked for a possible transmission.

In the illustrative example given in Section 6.5 as well as in the performance evaluation reported in Section 6.6, we assign priorities to nodes as follows. The priority of a node is computed taking into account:

1. the number of packets present in its queues, to avoid buffer saturation;
2. the sum for each data gathering tree of the number of packets its parent should receive in a cycle, to favor nodes with a high load;

3. and the classes of packets to be transmitted by the node in the current slot, to provide traffic differentiation if required by the application.

Furthermore, in order to obtain a strong traffic differentiation, we require that for any sink s the priority of any packet in a class i is higher than the priority of any packet in a class j with a strictly less importance degree ($j < i$). For simplicity sake, we assume that classes are ordered according to a non-decreasing importance degree. That is why, we define $prioClass_i$ for any class i and $prio_u$ for any node u as follows:

$$prioClass_i = \prod_{j<i}(1 + sinkRcv_j^2)$$

where $sinkRcv_j$ is the total number of packets that should be received by s for flows belonging to class j, where s is the sink associated with class j. By convention, $prioClass_1$=1.

$$prio_u = \sum_i [prioClass_i * \sum_{f \in i}(remPckt(u)_f * Rcv(Parent(u))_f)]$$

where $prioClass_i$ is the priority of class i to which a flow f present on node u belongs to, $remPckt_f$ means the number of packets of flow f the node u has in its buffer at the current iteration and $Rcv(Parent(u))_f$ is the total number of packets of flow f that the parent of node u has to receive in a cycle. Notice that in $prio_u$ two factors $prioClass_i$ and $Rcv(Parent(u))_f$ are static during the cycle whereas the factor $remPckt_f$ depends on the size of the buffer queue and hence depends on the slot considered.

6.4.2 Algorithm

The channel selection strategy given in the algorithm below is greedy but other strategies like Round Robin are also possible to achieve a better load balance.

Algorithm 2 MUSIKA algorithm

Input: cl_{max} traffic classes with their associated gathering trees, each node u has one available radio interface i_u, $nchannel$ channels, $Gen(u)_f$ packets of flow f to transmit and a set of conflicting nodes $Conflict(u)$.
Output: The multi-sink scheduling of nodes in the contention-free cycle
/* Initialization phase */
$\forall cl$, $prioClass_{cl} \leftarrow \prod_{cl'<cl}(1 + sinkRcv_{cl'}{}^2)$
for each node u **do**
  $remPckt(u)_f \leftarrow Gen(u)_f$ // number of packets initially present in the buffer of u
end for
$\forall u$, $prio_u \leftarrow \sum_i[prioClass_{cl} * \sum_{f \in cl}(remPckt(u)_f * Rcv(Parent(u))_f)]$
$t \leftarrow 0$ // current time slot
/* Scheduling phase */
while $\sum_f \sum_u remPckt_f$ **do** // there are packets to transmit
  $\forall u$, $i_u \leftarrow True$ // u has an available radio interface
  $\forall c = 1..nchannel$, $conflict_c \leftarrow \emptyset$ // initialize conflicting nodes on channel c
  $\forall cl$, $N_{cl} \leftarrow$ list of nodes having data to transmit and sorted according to their priorities in the class cl.
  $t \leftarrow t + 1$
  /* Assignment of slot t */
  while $\cup_{cl} N_{cl} \neq \emptyset$ **do**
    $Tx \leftarrow False, nChannelReached \leftarrow False$
    repeat
      Select the node v with the highest priority in $\cup_{cl} N_{cl}$
      $\forall cl$, $N_{cl} \leftarrow N_{cl} \setminus \{v\}$
      Select the flow f with the highest priority on node v
    until i_v and $i_{Parent(v)}$ // this node and its parent for flow f have an available interface
    $c \leftarrow 1$ // selected channel
    repeat
      if $v \notin conflict_c$ **then**
        Node v transmits in slot t on the channel c
        $remPckt(v)_f \leftarrow remPckt(v)_f - 1$
        $remPckt(Parent(v))_f \leftarrow remPckt(Parent(v))_f + 1$
        $i_v \leftarrow False$
        $i_{Parent(v)} \leftarrow False$
        $conflict_c \leftarrow conflict_c \cup Conflict(v)$
        Update $prio_v$ and $prio_{Parent(v)}$
        $Tx \leftarrow True$
      else
        if $c < nchannel$ **then**
          $c \leftarrow c + 1$ // change of selected channel
        else
          $nChannelReached \leftarrow True$
        end if
      end if
    until Tx || $nChannelReached$
  end while
end while

6.5 Illustrative example

The problem of optimal multichannel slot assignment is solved with the GPLK (GNU Linear Programming Kit) [GLPK 2011] solver based on the model presented in the section 6.3. We consider an illustrative multichannel network topology with two sinks (see Figures 6.1(a) and 6.1(b)). There are exactly two traffic types and one flow per traffic type, denoted f_1 and f_2. The optimal time needed for a raw data convergecast is computed considering a single traffic (f_1 or f_2) and then both types of traffic f_1 and f_2 differentiated by their destination sink. These traffic types can have different importance degrees from the application point of view. For simplicity sake, we also assume that each node generates one packet for each flow.

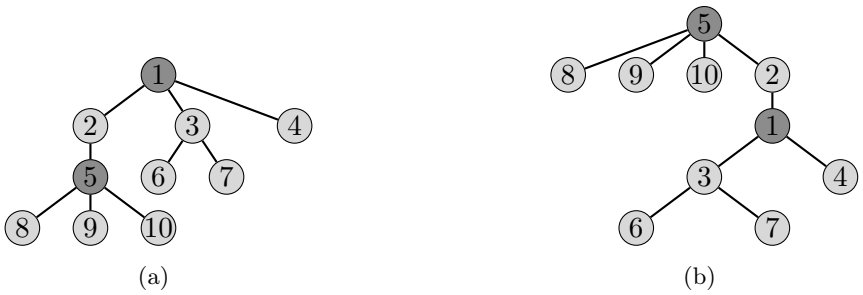


Figure 6.1: The two tree topologies of the network.

We use GLPK to compute first the minimum number of slots required by each flow taken separately. We obtain 9 slots for f_1 and 11 slots for f_2. For the minimum number of slots required by the two flows, GLPK gives 20 slots and the optimal schedule provided by GLPK is illustrated by Figure 6.2. In the figures representing schedules, we adopt the following convention: the transmission of a packet of flow f_1 is represented on a white background, whereas this of flow f_2 appears on a black background. The main number within the cell indicates the origin of the packet, whereas the index number indicates the channel used. Notice that only two channels are needed. In Figure 6.2, we observe that despite the parallelism of transmissions between the different flows, the optimal cycle length for the two flows is equal to the sum of the optimal cycle length for each flow taken separately.

When the flows have the same priority, MUSIKA provides the following schedule illustrated by Figure 6.3. We notice that the number of slots is optimal, equal to 20 slots. We observe also that the transmissions of flows f_1 and f_2 are interleaved.

When the flows have different priorities, MUSIKA gives the schedule illustrated by Figure 6.4. We notice that flow f_1 that belongs to the highest priority class completes in the slot 9, exactly as if it were alone in the WSN, unlike previously where it completed in slot 20.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$1 \rightarrow 2$		3_2				4_1						1_1			6_2			7_1		
$2 \rightarrow 1$					2_1		5_2		8_1				10_2							9_1
$2 \rightarrow 5$			3_2	2_2							4_2			1_2			6_2		7_2	
$3 \rightarrow 1$	3_2						6_2			3_1			6_2		7_1	7_1				
$4 \rightarrow 1$			4_1																4_2	
$5 \rightarrow 2$	5_1							8_2		10_2						9_1				
$6 \rightarrow 3$			6_2	6_2																
$7 \rightarrow 3$										7_2		7_2								
$8 \rightarrow 5$							8_1												8_2	
$9 \rightarrow 5$															9_2			9_1		
$10 \rightarrow 5$		10_2			10_2															

Figure 6.2: The optimal multi-sink slot assignment obtained with the GLPK solver for the illustrative example.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$1 \rightarrow 2$		1_1		3_1									6_1		7_1		4_1			
$2 \rightarrow 1$						2_1		5_1		8_1		9_1								10_1
$2 \rightarrow 5$	2_1		1_1		3_2									6_1		7_1		4_1		
$3 \rightarrow 1$	3_2		6_2		3_1		6_1		7_1					7_2						
$4 \rightarrow 1$											4_1					4_2				
$5 \rightarrow 2$							5_1		8_1		9_1								10_1	
$6 \rightarrow 3$		6_1		6_1																
$7 \rightarrow 3$						7_1		7_1												
$8 \rightarrow 5$		8_1						8_2												
$9 \rightarrow 5$				9_1						9_2										
$10 \rightarrow 5$						10_2						10_2								

Figure 6.3: The multi-sink slot assignment obtained with MUSIKA for the illustrative example with flows f_1 and f_2 having the same priority.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$1 \rightarrow 2$											1_1		3_1		6_1		7_1		4_1	
$2 \rightarrow 1$	2_1		5_1		8_1		9_1		10_1											
$2 \rightarrow 5$										2_2		1_1		3_1		6_1		7_1		4_1
$3 \rightarrow 1$		3_1		6_1		7_1				3_1		6_2		7_2						
$4 \rightarrow 1$								4_1								4_2				
$5 \rightarrow 2$		5_1		8_1		9_1		10_1												
$6 \rightarrow 3$	6_1				6_1															
$7 \rightarrow 3$			7_1				7_1													
$8 \rightarrow 5$	8_2						8_2													
$9 \rightarrow 5$			9_2						9_2											
$10 \rightarrow 5$					10_2						10_1									

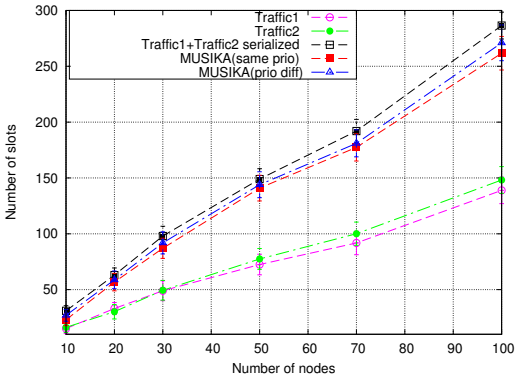
Figure 6.4: The multi-sink slot assignment obtained with MUSIKA with flow f_2 represented in black belongs to a class of less importance degree than flow f_1 represented in white.

6.6 Performance evaluation

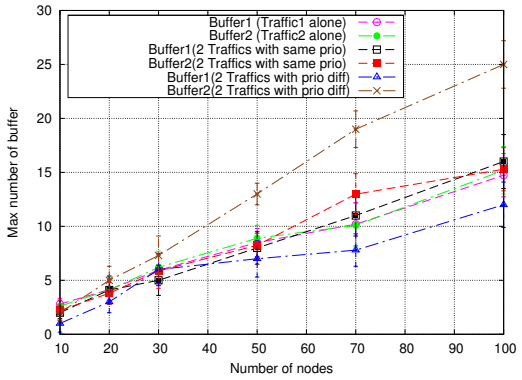
We use a simulation tool based on GNU Octave [Octave] to evaluate the performances of MUSIKA in various topologies of WSNs. The number of nodes varies from 10 to 100. All nodes have a single radio interface. We assume that the only links are those belonging to the tree. In these simulations, $Conflict(u)$ is the set of one-hop and two-hop neighbors of

u, for any node u. For each data gathering tree considered, the number of children per node is less than or equal to 3. Each result depicted in a curve represented in Figures 6.5(a) to Figure 6.5(c) is the average of 20 simulation runs.

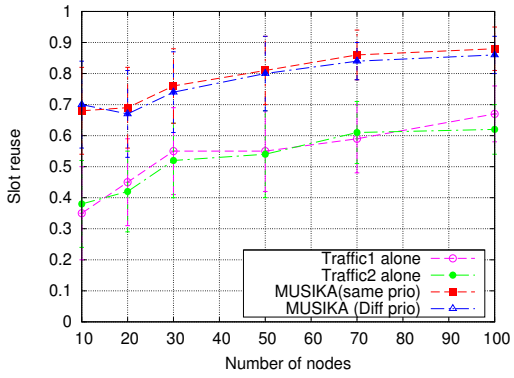
In the first series of experiments, we evaluate the average number of slots needed to complete convergecast. We distinguish two cases: both traffics have the same priority and a traffic has a higher priority than the other. Results are depicted in Figure 6.5(a). It appears that the number of slots required by MUSIKA in both cases (same and different priorities) is less than the number of slots when traffic 1 and traffic 2 are serialized: traffic 1 served before traffic 2. MUSIKA reduces the number of slots by first optimizing the scheduling of the first traffic and then applying spatial reuse to schedule the second traffic. For example in the 100 nodes configuration, as depicted in Figure 6.5(a), MUSIKA needs only 261 slots while a serial schedule of the two flows needs 288 slots, providing a gain of about 10%.



(a) Number of required slots



(b) Maximum buffer size



(c) Slot reuse ratio

Figure 6.5: MUSIKA performances.

In the second series of experiments, we also evaluate the maximum number of buffers required in a node during a contention-free cycle. Figure 6.5(b) shows that when both traffics have the same priority, MUSIKA requires a number of buffers close to the number of buffers required when only one traffic is present. This can be explained by the basic priority of nodes

used in MUSIKA which favors nodes having longer buffer queues to transmit.

In the third series of experiments, we consider the slot reuse ratio defined by the ratio of slots used by at least two transmissions. As illustrated by Figure 6.5(c), MUSIKA achieves the higher slot reuse ratio in the two cases of traffic priority. As shown in Figure6.5(c), for 100 nodes the slot reuse ratio is equal to 0.88 (two traffics with the same priority) while this ratio is equal to 0.68 when there is only one traffic. That can be explained by the fact that MUSIKA takes benefit of the spatial reuse concept when computing the schedules. Undoubtedly, the slot reuse ratio significantly reduces the end-to-end latency without a penalty in energy efficiency.

6.7 Conclusion

Typical convergecast scenarios for large-scale WSNs contain multiple sources and multiple sinks. Therefore, we tackled in this chapter, the problem of minimizing the time needed to complete convergecast in multi-sink WSNs. We have identified two main reasons for the existence of several sinks: on the one hand redundancy of sinks improves robustness of data gathering and on the other hand, different application functionalities may be distributed on the sinks, explaining why they may have different importance degrees.

Our contribution in this chapter is twofold. First, we aimed at deriving collision-free schedules for raw data convergecast with minimum latency in multi-sink multichannel WSNs. To achieve this objective, we formulate the problem as a linear programming problem, aiming at deriving the smallest cycle length. We used GLPK as a tool to solve the formulated multi-sink multichannel scheduling optimization problem. Second, we propose the MUSIKA algorithm to obtain a collision-free schedule with a minimized frame length. This algorithm provides traffic differentiation if required by the application to reflect different importance degrees of traffic. From the simulation results, we conclude that MUSIKA shows its merit by taking advantage of spatial reuse to assign any slot to non-conflicting transmitters in both traffics, thus reducing the cycle length. Furthermore, the maximum number of buffers needed on a node is optimized with MUSIKA.

Although we have well studied raw convergecast in both theoretical and practical aspect in previous chapters, the question of what to do if retransmissions or temporary changes in application needs, is unanswered. Furthermore, all the previous algorithms for generating multichannel collision-free schedules in the first part are centralized. Indeed, the sink has to compute the schedule and disseminate it to the sensors. Once all the sensor nodes receive the schedule, they work according to the schedule. Since topology changes occur commonly in WSNs such as node failures, the sink has to gather new topology information from the network, recompute a schedule and disseminate it frequently. This situation makes centralized algorithms inefficient in large-scale WSNs. Thus, the target of the next part will be the proposal of adaptive conflict-free schedules and distributed scheduling algorithms for data convergecast.

Part III

Adaptivity and Scalability of Joint Time Slot and Channel Assignment

Chapter 7

An Adaptive Strategy for an Optimized Collision-Free Slot Assignment in Multichannel Wireless Sensor Networks

Contents

7.1 Introduction . . . 107
7.2 State of the art . . . 108
7.3 Adaptive multichannel slot assignment . . . 110
7.3.1 Definitions . . . 110
7.3.2 Assumptions . . . 111
7.3.3 Network model . . . 111
7.3.4 Theoretical bounds on the number of extra slots for a raw data convergecast 113
7.3.5 AMSA: Proposed solution . . . 116
7.3.6 Illustrative example . . . 119
7.4 Performance evaluation . . . 121
7.4.1 Retransmission oriented experiments . . . 122
7.4.2 Temporary change in the application needs-oriented experiments . . . 126
7.5 Conclusion . . . 134

7.1 Introduction

In Part II of this manuscript, we focused on the joint time slot and channel assignment that minimizes the data gathering delays. Yet, in many real deployments, the WSN encounters dynamic demands of transmissions (e.g., alarms, temporary additional traffic). Hence, the design of an adaptive slot assignment algorithm preserving the initial optimized assignment, is crucial. This algorithm should adapt to:

- Retransmission of a message that has not been acknowledged at the MAC layer.
- Temporary change in the application needs due to alarms, for instance. Alarms are associated with strong delay requirements and must be reliably delivered to the sink. Several possibilities exist:
  - if a slot was assigned to the sender node, the alarm is sent first, taking the place of the regular data. This regular data will then be sent in an additional slot, granted by the adaptive solution.
  - if no slot was assigned to the sender node; in the worst case, the alarm is sent in the next cycle following the control message requesting the alarm transfer.

Therefore, we would like to make the slot assignment more flexible and able to adapt to application and environment variability. In other words, we would like to benefit from the intrinsic adaptivity of CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) in a deterministic slotted scheme.

We have to take into account several constraints, particularly in large-scale WSNs. First, the computation time is constrained by the computing power of the embedded node in charge of the slot assignment, usually the sink. Second, due to the small payload size in WSNs based on the IEEE 802.15.4 standard, the sink may need to transmit several messages to disseminate the amount of data defining the slot assignment. Third, to prolong the network lifetime, node residual energy must be saved by transmitting and receiving fewer messages. That is why **we propose a solution based on an incremental technique**. The slot assignment is built initially and totally rebuilt only when there are non transient changes in transmission needs. Indeed, it is preferable to transmit only the differential data corresponding to a temporary slot assignment.

7.2 State of the art

Many existing time slot assignment protocols for WSNs operate independently of the dynamic application demands, e.g., retransmissions, alarms, additional injected traffic. This is, however, a particular challenge in systems with strong reliability or alarm-based systems. In this section, we present some studies that deal with the adaptivity of time slot assignments. While several studies have tackled this concern in mono-channel WSNs, a comprehensive technique is still lacking in multichannel WSNs.

In [Miao 2008], the authors present EP-TDMA, an evolutionary dynamic slots assignment algorithm that operates in two phases. More precisely, the cycle consists of two steps: (1) a control phase, which is subdivided into a claim subphase and a response subphase with CSMA/CA competition mechanism, and (2) an information phase, where data transmission takes place. First, each phase consists of N slots, where N is the number of nodes in the network. Therefore, each node has its own slot. A node is said to be active if it has a packet to transmit in its slot assigned . During the claim phase, active nodes add information in claim packets to announce their need for more slots if they encounter a high workload. Then, each node collects its neighbors' claim packets and transmits a response packet in a response

slot. Therefore, by exchanging response packets, any node, u, can compute the slots it can compete for (i.e. slots during the information phase) and the set of neighbors up to two hops away. Node u cannot use any slot already used by any other node in this set. To solve the slot competition between nodes, any node gets a random priority for each slot, except its own slot, for which it has the highest priority. After the exchange of claim and response packets, the nodes determine active nodes and active nodes' traffic within two hops. Then, each node knows all the slots it can obtain, *i.e.*, the slots for which it has the highest priority. This technique allows active nodes to obtain additional slots without the risk of collisions. Nevertheless, under heavy traffic, the contention during the control phase increases, resulting in a low delivery ratio.

The authors of [Yackovich 2011] enhance TDMA-ASAP [Gobriel 2009b] by minimizing the latency of the report of any event occurring during the cycle, assuming that events are uniformly distributed in a cycle. The enhancement consists in:

- A slot spreading algorithm: Nodes in TDMA-ASAP steal only from adjacent or nearby slots. The proposed algorithm spreads out children slots evenly in the schedule.

- Stealing both in upstream and downstream communication: The authors assume that any cycle consists of upstream slots and optional downstream slots spread throughout the cycle. Since downstream slots are not necessary in every cycle, they can be stolen by upstream communications. If the children do not receive a control message from their parents, they treat the downstream slot reserved for their parent as a stealable slot.

In [Kanzaki 2009], Kanzaki *et al.* propose Adaptive Slot Assignment Protocol with Slot Migration (ASAP/SM), which is an extension of their previous work on Extended Adaptive Slot Assignment Protocol (E-ASAP). This latter is not traffic-aware, because nodes that have heavy traffic do not have enough assigned slots. ASAP/SM obeys E-ASAP rules for slot allocation. The decision of migrating to node N_j a slot assigned to node N_i depends on certain conditions: the amount of traffic at node N_i (channel requirements), channel utilization (number of slots assigned to N_i divided by the frame length) and the gap value (the difference between two nodes, N_i and N_j, in terms of channel utilization and traffic load). The cycle of slot migration is called an "update cycle". During this cycle, a node begins by calculating the gap value of its neighbors. Then, it sets priorities according to gap values: the larger the gap is, the higher the priority. For a node N_i, the target node (*i.e.*, the first node to be stolen) is its neighbor with the highest priority. Node N_i should select a slot, already allocated to its neighbor, to become its own. If node N_i does not find a slot that can be migrated, the next node with the highest priority is set as the next target node. In addition, to avoid slot migration oscillations, which lead to more control traffic, the authors propose a mechanism to cancel slot migration according to certain criteria.

Tselishchev *et al.* [Tselishchev 2011] target retransmission strategy for body area networks taking into account energy criteria and the temporal variation in the wireless links. The authors consider a single hop network, where a hub communicates directly with n sensors. The time is divided into rounds of m slots ($m > n$). In each round, the first n slots are

dedicated to regular transmissions of packets. Hence, $(m - n)$ slots are left as sparse. When a sensor fails to transmit a packet, it is added to a list called the "retransmission eligibility list". This list contains nodes that have backlogged packets and an energy level greater than a specific energy threshold. Then, a node is randomly selected from the list and assigned a sparse slot. The process is repeated until the list becomes empty. They propose a technique called "Flip + Spread", which enhances the successful transmission ratio in a round that includes a retransmission. Assuming that the first transmission of a message m fails during the first round, this technique places the second transmission attempt of m at the end of the second round. The first transmission attempt of the next message after m is placed in the second round in the middle of the first transmission (previous round) and the second transmission attempt of m. Therefore, transmissions are spread over two rounds, provided that the number of nodes requesting a retransmission in any round is less than $m - n$. The shortcoming of this work is that the authors restrict their study to single hop networks. It is generally assumed that WSNs are deployed over a large area and, hence, multi-hop convergecast structure is usually adopted.

As a conclusion, we observe that none of the solutions studied previously guarantees the assignment of slots along the path of the raw data convergecast tree. Hence, there is no guarantee that the data, including those corresponding to the additional demands, will be gathered in a single frame. We propose AMSA to meet this requirement, if this is allowed by the inactivity period length. Furthermore, we notice that changes in transmission needs are handled either by a new slot assignment or by an incremental technique. For the reasons given in the previous section, we prefer the latter for transient changes.

7.3 Adaptive multichannel slot assignment

As seen in Section 7.2, many existing time slot assignments lack flexibility. They are unable to adapt to additional demands (e.g., retransmission, new application needs, alarms). That is why in this section, we are looking for an incremental technique that is able to update a given optimized time slot assignment to meet new application needs. This technique should preserve the initial time slot assignment, provided, for instance, by MODESA, and minimize the number of extra slots added to the initial slot assignment.

7.3.1 Definitions

First, we introduce some extra definitions:

Definition 7. ***Ordinary node:*** *a node is said ordinary if and only if is is not a sink.*

Definition 8. ***Primary schedule:*** *is the initial optimized time slot assignment obtained with MODESA; it includes only slots needed by regular messages. These slots are said to be regular. Consequently, the number of regular slots granted to any ordinary node is equal to the sum of its initial demand and the initial demands of all its descendants.*

Definition 9. ***Bonus slot:*** *is a temporary slot assigned after a change notification (e.g., temporary change in the application need, retransmission,* etc.*).*

Definition 10. ***Secondary schedule:*** *is the optimized assignment of bonus slots obtained with AMSA, the incremental time slot assignment algorithm we propose in this chapter and describe in Section 7.3.5.*

Definition 11. ***Slot path:*** *is a sequence of slots that allows the transmission of a message from any given ordinary node to the sink.*

Definition 12. ***Extra slot:*** *is a bonus slot that is appended at the end of the primary schedule. Any extra slot increases the length of the activity period.*

Definition 13. ***Complementary schedule:*** *is the optimized time slot assignment obtained with MODESA taking into account only the requests of bonus slots. It starts at the end of the primary schedule.*

7.3.2 Assumptions

We also introduce some additional assumptions which are required by the adaptive multichannel slot assignment.

Assumption 8. ***Service differentiation policy:*** *Each node transmits its messages according to their priority. Different priority levels can be distinguished:*

- *alarms have the highest priority,*
- *retransmissions of regular messages have medium priority,*
- *regular messages have the lowest priority.*

Assumption 9. ***Computation time for bonus slot assignment:*** *We assume that the computation time for bonus slot assignment plus the time of the extra slots is less than the duration of inactivity.*

7.3.3 Network model

The network is formalized as a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices representing the nodes of the network and $\mathcal{E}$ is the set of edges representing the communication links between the nodes.

Let $\mathcal{V} = \mathcal{V}_n \bigcup \mathcal{V}_s$, where $\mathcal{V}_n$ is the set of ordinary nodes and $\mathcal{V}_s$ represents the set of sinks, with $\mathcal{V}_n \bigcap \mathcal{V}_s = \emptyset$. For each node, $v \in \mathcal{V}$, we define $Conflict(v)$, the set of conflicting nodes that interfere with v when transmitting on the same channel. Moreover, let i_v denote the number of physical interfaces available at any node, v. For any ordinary node, $n \in \mathcal{V}_n$, let r_n correspond to the number of bonus slots requested by n to transmit the additional packets it generates.

Let $\mathcal{E}^+(v)$ denote the set of links through which a node, v, can transmit. Let $\mathcal{E}^-(v)$ be the set of links through which a node, v, can receive. For any link, $e \in \mathcal{E}$, let $f_{e,n}$ denote the number of slots needed to transmit over the link, e, the additional packets generated by node, n.

Let $\mathcal{C}$ be the set of channels usable for any transmission. The collision-free period is composed of slots, t, in the interval, $[1; T_{max}]$, where T_{max} denotes an upper bound of the frame length expressed in a number of slots. This bound is reached when all packets are sent sequentially on the same channel. We then have: $T_{max} = \sum_n \sum_e f_{e,n} * depth_n$, where $depth_n$ is the depth of node, n, in the data gathering tree.

Let the parameter $A_{e,c,t}$ correspond to the activity of a link, e, on the channel, c, in the time slot, t, in the primary schedule, *i.e.*, $A_{e,c,t} = 1$, if and only if there is a transmission of a packet on the link, e, on the channel, c, in the time slot, t, in the primary schedule, and $A_{e,c,t} = 0$, otherwise. This binary parameter is given by the MODESA algorithm.

We define $b_{e,c,t}$ as bonus slot assignment for a link, e, on the channel, c, in the time slot, t, *i.e.*, $b_{e,c,t} = 1$, if and only if there is a transmission of a packet on the link, e, on the channel, c, in the time slot, t, in the bonus assignment, and $b_{e,c,t} = 0$, otherwise.

Furthermore, let u_t be the use of a slot, t, in other words, $u_t = 1$ means that there is at least one link activity (activity in the primary or secondary schedule) on at least one channel in the slot, t, and $u_t = 0$ denotes an empty slot.

Table 7.1 summarizes the inputs and variables of our model.

Inputs	
$\mathcal{E}$	set of links in the topology
$\mathcal{V}$	set of nodes in the topology
$\mathcal{C}$	set of available channels
i_v	number of interfaces of the node, v
$Conflict(v)$	set of nodes conflicting with v when transmitting on the same channel in the same time slot
r_n	number of bonus slots requested by node, n
$A_{e,c,t}$	activity of the link e in time slot, t, on channel, c,
Variables	
u_t	utility of slot, t
$f_{e,n}$	number of slots needed to transmit over link, e, the additional packets generated by node, n
$b_{e,c,t}$	assignment of bonus slot, t, to link, e, on channel, c,

Table 7.1: Inputs and variables of the model.

The objective is to minimize the number of slots, $t \leq T_{max}$:

$$min \sum_{t \leq T_{max}} u_t$$

with the following constraints:

$$\forall c \in \mathcal{C}, \forall e \in \mathcal{E}, t \leq T_{max}, \quad b_{e,c,t} + A_{e,c,t} \leq u_t \tag{7.1}$$

Constraint 7.1 binds the use of a time slot to at least the activity of one link on any channel in this slot in the primary schedule or the secondary schedule.

$$\forall v \in \mathcal{V}, \forall e \in \mathcal{E}^+(v), \forall w \in Conflict(v), \forall e' \in \mathcal{E}^+(w), \forall c \in \mathcal{C}, t \leq T_{max}, \quad b_{e,c,t} + A_{e',c,t} \leq 1 \tag{7.2}$$

Constraint 7.2 guarantees that a node cannot transmit while one of its conflicting nodes is already scheduled in the same slot on the same channel in the primary schedule.

$$\forall v \in \mathcal{V}, \forall e \in \mathcal{E}^+(v), \forall w \in Conflict(v), \forall e' \in \mathcal{E}^+(w), \forall c \in \mathcal{C}, t \leq T_{max}, \quad b_{e,c,t} + b_{e',c,t} \leq 1 \tag{7.3}$$

Constraint 7.3 ensures that for the bonus slot assignment, two conflicting nodes do not transmit on the same channel in the same time slot.

$$\forall v \in \mathcal{V}, t \leq T_{max}, \quad \sum_{c\in\mathcal{C}} \sum_{e\in\mathcal{E}^+(v)} b_{e,c,t} + \sum_{c\in\mathcal{C}} \sum_{e'\in\mathcal{E}^-(v)} b_{e',c,t} + \sum_{c\in\mathcal{C}} \sum_{e''\in\mathcal{E}^+(v)} A_{e'',c,t} + \sum_{c\in\mathcal{C}} \sum_{e'''\in\mathcal{E}^-(v)} A_{e''',c,t} \leq i_v \tag{7.4}$$

Constraint 7.4 limits the number of simultaneous communications for a node, during the primary schedule and the bonus assignment, to its number of interfaces.

$$\forall e \in \mathcal{E}, \quad \sum_{n\in\mathcal{V}_n} f_{e,n} = \sum_{c\in\mathcal{C}} \sum_{t\leq T_{max}} b_{e,c,t} \tag{7.5}$$

Constraint 7.5 ensures the mapping between the activities on all channels and the packets sent on links.

$$\forall n \in \mathcal{V}_n, \quad \sum_{e\in\mathcal{E}^+(n)} f_{e,n} = r_n \tag{7.6}$$

$$\forall i \in \mathcal{V}_n, \quad \sum_{n\in\mathcal{V}_n} \sum_{e\in\mathcal{E}^+(i)} f_{e,n} = r_i + \sum_{n\in\mathcal{V}_n} \sum_{e\in\mathcal{E}^-(i)} f_{e,n} \tag{7.7}$$

$$\forall n \in \mathcal{V}_n, \quad \sum_{s\in\mathcal{V}_s} \sum_{e\in\mathcal{E}^-(s)} f_{e,n} = r_n \tag{7.8}$$

Constraints 7.6 to 7.8 express the conservation of messages, respectively, at the nodes requesting a bonus slot, at intermediate nodes and at the sinks.

$$\forall n \in \mathcal{V}_n, t \leq T_{max}, \quad \sum_{c\in\mathcal{C}} \sum_{e\in\mathcal{E}^+(n)} b_{e,c,t} \leq r_n + \sum_{c\in\mathcal{C}} \sum_{e\in\mathcal{E}^-(n)} \sum_{t'\in\{1..t-1\}} b_{e,c,t'} - \sum_{c\in\mathcal{C}} \sum_{e\in\mathcal{E}^+(n)} \sum_{t'\in\{1..t-1\}} b_{e,c,t'} \tag{7.9}$$

Constraint 7.9 makes certain that a packet is generated or received by a node before being transmitted or forwarded.

Notice that this model can be applied to compute the primary schedule by setting $A_{e,c,t} = 0 \quad \forall c \in \mathcal{C}, \forall e \in \mathcal{E}, t \leq T_{max}$

7.3.4 Theoretical bounds on the number of extra slots for a raw data convergecast

We now provide lower and upper bounds for the number of extra slots added after the regular ones by any incremental solution. This should preserve the primary schedule and ensure that the data gathering, including the additional demands, is performed in a single frame.

Property 1. *The minimum number of extra slots is given by the difference between, on the one hand, the optimal slot assignment meeting for each node the sum of its initial and the new demands and, on the other hand, the number of regular slots.*

Proof. The number of slots required by the incremental algorithm is the sum of regular slots and extra slots. In the best case, this sum is equal to the number of slots obtained by the optimal algorithm to schedule both initial and new demands. Hence, the minimum number of extra slots is the difference between the number of slots given by the optimal algorithm and the number of regular slots. □

Property 2. *The maximum number of extra slots is given by* $\sum_{u \text{ requesting node}} depth_u \times r_u$*, where* $depth_u$ *is the depth of any ordinary node,* u*, in the convergecast tree and* r_u *is the number of bonus slots requested by* u*.*

Proof. The maximum number of extra slots added to the regular ones corresponds to the worst case, which occurs when there is no parallelism between the slot paths granted, corresponding to the additional demands. It is given by $\sum_{u \text{ requesting node}} depth_u \times r_u$, where $depth_u$ is the depth of any ordinary node, u, in the convergecast tree; it is also the length of the path from this node to the sink, and r_u is the number of bonus slots requested by u. □

We can improve this upper bound, by taking into account some parallelism in the transmissions. We then get:

Property 3. *The maximum number of extra slots is given by* $MaxDepth + 2\sum_{u \text{ requesting node}} r_u - 2F_{Maxdepth}$*, where* $MaxDepth$ *is the maximum depth in the convergecast tree of any ordinary node requesting bonus slots,* r_u *is the number of bonus slots requested by any ordinary node,* u*, and* $F_{Maxdepth}$ *is defined as follows:*

$$\begin{aligned} F_{Maxdepth} &= \sum_{k=1}^{maxDepth/2} 1_{2k} \qquad \text{if } Maxdepth \text{ is even,} \\ &= \sum_{k=0}^{\lfloor maxDepth/2 \rfloor} 1_{2k+1} \qquad \text{otherwise,} \end{aligned}$$

with $1_k = 1$*, if and only if there is at least one node of depth* k *requesting bonus slots.*

Proof. We now build a schedule for all the requests for bonus slots. Like AMSA, this schedule assigns slots per path from the requesting node to the sink. Let u be the node with the greatest depth in the convergecast tree requesting bonus slots. Let $Maxdepth$ be its depth. If scheduled first, node u, will require $MaxDepth$ extra slots for the first bonus slot requested.

Let us consider Figure 7.1. Nodes 4, 10 and 12, depicted in orange color, require bonus slots. $Maxdepth$ is equal to the depth of node 12. For instance, in Table 7.2, a node at $MaxDepth = 6$ (node 12 for example) will require six extra slots (see the grey slot path). For the next bonus slots required by u, if any, each of them will require two extra slots (see the black slot path in Figure 7.2 ending at slot 8). Indeed, the next transmission of u cannot be scheduled in the next slot, where the parent of u is transmitting. Similarly, all the bonus slots requested by the other nodes will need two additional extra slots. Hence, we get $MaxDepth + 2\sum_{w \text{ requesting node}} r_w$ extra slots. We can also notice that in the slot,

where u is scheduled, a node v (for example 10), of depth, $Maxdepth - 2$, can be scheduled simultaneously, but on another channel (assuming that the links in the convergecast tree are the only topology links). Applying this property recursively, we can then schedule in the $MaxDepth$ first slots, one bonus request by a node of depth k, with k varying from $MaxDepth$ to two if $MaxDepth$ is even and from $MaxDepth$ to one, otherwise. Hence, we finally get the maximum number of extra slots equal to $MaxDepth + 2\sum_{u\ requesting\ node} r_u - 2F_{Maxdepth}$, with:

$$\begin{aligned} F_{Maxdepth} &= \sum_{k=1}^{maxDepth/2} 1_{2k} \quad \text{if } Maxdepth \text{ is even,} \\ &= \sum_{k=0}^{\lfloor maxDepth/2 \rfloor} 1_{2k+1} \quad \text{otherwise,} \end{aligned}$$

□

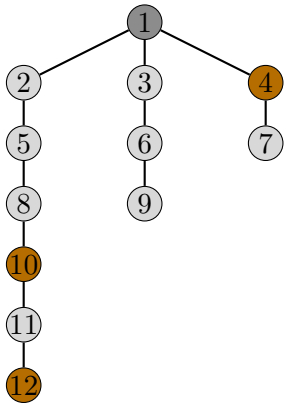


Figure 7.1: Illustrative example of the upper bound.

Table 7.2: Extra slots scheduling when $Maxdepth = 6$

Depth \ Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-	1_2	-	1_4	-	1_6	-	2_6	-	3_6	-	1_5	-	2_5
2	1_2	-	1_4	-	1_6	-	2_6	-	3_6	-	1_5	-	2_5	-
3	-	1_4	-	1_6	-	2_6	-	3_6	-	1_5	-	2_5		
4	1_4	-	1_6	-	2_6	-	3_6	-	1_5	-	2_5	-	-	-
5	-	1_6	-	2_6	-	3_6	-	1_5	-	2_5	-	-	-	-
6	1_6	-	2_6	-	3_6	-	-	-	-	-	-	-	-	-

Notice that this bound does not take into account the possible parallelism between simultaneous transmissions of nodes belonging to the same depth. AMSA takes advantage of this parallelism.

The performance evaluation reported in Section 7.4 will position AMSA with regard to an intermediate solution, where an optimized time slot assignment is computed by MODESA from the bonus slots requests and appended at the end of the primary schedule.

7.3.5 AMSA: Proposed solution

7.3.5.1 Principles

The incremental solution we propose is based on the following principles. Each node that detects a change in its transmission needs (e.g., temporary change in the application need, retransmission, *etc.*), notifies the slot manager, usually the sink, by means of a control message. This control message is sent in the control part of the frame and forwarded to the parent in the convergecast tree, until it reaches the sink. AMSA builds the secondary schedule (see Algorithm 3). This is computed by the sink after having received the bonus requests sent, according to Algorithm 4. The sink tries to insert bonus slots into the primary schedule without increasing its length to meet the additional needs. If that is impossible, it takes new slots in the inactivity period. The sink transmits this new slot assignment downstream the tree, as the primary slot assignment was transmitted.

When a node receives the new slot assignment, it uses (indifferently) regular and bonus slots to transmit its messages according to the service differentiation policy expressed in Assumption 8. Notice that the bonus slots are allocated temporarily and are only valid for one data gathering frame. The associated use case corresponds to alarm transmissions or message retransmissions. If the application needs increase or decrease for a longer time, the new primary schedule is recomputed with MODESA.

7.3.5.2 AMSA algorithm

The AMSA algorithm has two parts: one part is run by the sink in charge of slot assignment, whereas the other part is run by ordinary nodes. With AMSA, the sink processes the bonus requests from origin nodes one by one, assigning one bonus slot to the whole path to the sink, until all the bonus requests have been met. AMSA takes as input the primary schedule, the bonus requests and the conflicting nodes. It completes the primary assignment by adding one or several tuples $(slotnumber, sender, receiver, channel)$, applying the following rules:

1. Any node requesting a bonus slot has a static priority, which is equal to $depth_u * r_u$, where $depth_u$ depicts the depth of a node u in the convergecast tree and r_u is the number of requested bonus slots (see line 6 of Algorithm 3). This priority favors nodes requesting a longer slot path or a higher number of bonus slots. The goal is to minimize the number of extra slots. The length of a slot path is given by the depth of the requesting node. Since AMSA allocates slots per slot path, scheduling the longer path first helps AMSA to complete the schedule earlier.

2. Nodes having bonus requests are sorted according to their priorities: the node with the highest priority is selected first (see line 7 of Algorithm 3).

3. The sink serves the bonus requests from the selected node, assigning one bonus slot to the whole path to the sink (see lines 9 to 34 of Algorithm 3).

4. To be allowed to transmit in a slot, both the selected node node and its parent should have an available radio interface (see line 12 of Algorithm 3). Consequently, AMSA searches for the first slot where this condition is met.

5. AMSA searches for the first channel where this node does not conflict with the already transmitting nodes on the same channel. Hence, a node is scheduled in the earliest possible slot (see lines 13 to 22 of Algorithm 3).

6. Any ordinary node, u, maintains a counter, r_u, that corresponds to the number of bonus slots it will request: see Algorithm 4.

Algorithm 3 AMSA algorithm for the sink

1: **Input:** *nchannel* channels; a spanning tree, T, where each node, u, has a depth, $depth_u$, a set of conflicting nodes, $Conflict(u)$, and a request of bonus slots, r_u; a primary slot assignment with for each slot, t, the number of available radio interfaces for each node, u.
2: **Output:** The bonus slot schedule in the data gathering frame
3:
4: $\mathcal{N} \leftarrow$ the set of nodes having requested bonus slots
5: **while** $\mathcal{N} \neq \emptyset$ **do** // there are bonus slots to assign
6: Sort $\mathcal{N}$ according to the decreasing priority of nodes, with $prio_u \leftarrow depth_u \times r_u$
7: $u \leftarrow first(\mathcal{N})$ // assign slots to a path starting with node u
8: $t \leftarrow 1$ // starts from the first time slot
9: **while** u is not the sink **do** // assign a slot to node u
10: $ChannelFound \leftarrow False$
11: **while** $Not(ChannelFound)$ && $t \leq MaxSlot$ **do**
12: **if** the node, u, and its parent have an available interface in slot, t **then**
13: $c \leftarrow 1$ // selected channel
14: **while** $Not(ChannelFound)$ && $c \leq nchannel$ **do**
15: **if** node u does not conflict with nodes already scheduled on channel c in time slot t **then**
16: Assign time slot t to u on channel c
17: Update the request of node u and its parent
18: Update the priority of node u and its parent
19: $ChannelFound \leftarrow True$
20: **end if**
21: $c \leftarrow c + 1$
22: **end while** // channel
23: **end if** // available interface
24: $t \leftarrow t + 1$ // try another slot
25: **end while** // Slot
26: **if** the request of u has been totally served || $Not(ChannelFound)$ **then**
27: **if** $u \in \mathcal{N}$ **then**
28: $\mathcal{N} \leftarrow \mathcal{N} \setminus \{u\}$
29: **end if**
30: **end if**
31: **if** $ChannelFound$ **then**
32: $u \leftarrow Parent(u)$ // continue the assignment of the slot path considering the parent of u
33: **end if**
34: **end while** // slot path
35: **end while** // no pending demand

Algorithm 4 AMSA algorithm for any ordinary node, u

Input: The additional application demand, Ad_u
Output: The number of bonus slots requested, r_u

$r_u \leftarrow Ad_u$
for all slots t where u transmits **do**
  if t is a regular slot && no acknowledgment received **then**
    $r_u = r_u + 1$
  else
    if t is a bonus slot && acknowledgment received **then**
      $r_u = r_u - 1$
    end if
  end if
  if t is the last slot where u transmits **then**
    if r_u **then**
      Send a BonusRequest message with r_u in the bonus field
    end if
  end if
end for

7.3.5.3 Discussions

AMSA brings the following advantages:

- *spatial reuse:* AMSA does not systematically require additional slots, due to its opportunistic behavior. Indeed, it takes advantage of spatial reuse to fill the slots with the new demands and, if that is impossible, adds a minimum number of slots. In both cases, AMSA ensures that (1) the number of available radio interfaces allows the transmission from the node considered to its parent and (2) no two conflicting nodes will transmit in the same slot on the same channel.

- *a unique algorithm that can be used both for retransmissions at the MAC level and new transmission needs at the application level.* The same algorithm is able to adapt to both application or MAC changes in their transmission needs. Furthermore, AMSA assigns a whole slot path. Indeed, each time a node requests an additional slot, the whole slot path corresponding to the slot sequence needed to reach the sink will, if possible, be granted.

- *optimized retransmissions and a simpler implementation:* on the one hand, we notice that with AMSA, only the sequence of slots starting with the transmitter that has not received the acknowledgment is allocated. On the other hand, the implementation is made simpler, because on any node, at any time, there is at most one pending message waiting for its acknowledgment.

- *an energy efficient convergecast:* Firstly, it minimizes the number of slots which is crucial from an energy point of view, as it allows nodes to sleep to save energy. On

the other hand, conflict avoidance on the bonus and primary slots avoids collisions and, hence, contributes also to saving energy.

Furthermore, we compared the overhead induced by AMSA and MODESA regarding the number of messages and the complexity. Our proposed incremental technique requires fewer messages to provide the additional schedule: AMSA requires fewer messages than MODESA recalculated: only nodes that are involved in the assignment of bonus slots are destinations of the message giving the new schedule, unlike MODESA, where all nodes are involved. Regarding complexity, Table 7.3 summarizes the main differences between AMSA and MODESA.

Table 7.3: Comparative table between AMSA and MODESA recalculated.

		AMSA	**MODESA recalculated**
Priority	nodes involved	only nodes requesting bonus slots	all nodes
	when it is computed	once at the beginning of the algo	at the beginning of any slot
Computation of the slot assigned to the highest priority node		any slot from the first one up to the current one	the current one
Number of transmissions to schedule		$\sum_{u \neq sink} depth_u * r_u$	$\sum_{u \neq sink} depth_u * (d_u + r_u)$

7.3.6 Illustrative example

In this section, we provide two scenarios of bonus slot assignment. Figure 7.2 presents the topology of the network studied in our example. This network is composed of 10 nodes, including the sink (node 1). The sink has two radio interfaces. In both scenarios, there are two channels available at each node.

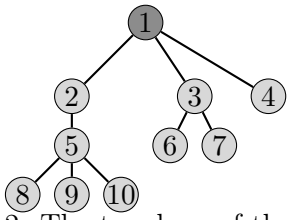


Figure 7.2: The topology of the network.

The optimal primary schedule given by MODESA is depicted in Figure 7.3. This assignment has a length of nine slots and corresponds to a primary demand of only one packet generated at each node except the sink. Slots are represented by a column. The notation, 2_1, means that node 2 transmits data to its parent on channel 1.

Slot / $tx \rightarrow rx$	1	2	3	4	5	6	7	8	9
$2 \rightarrow 1$	2_1		5_1		8_1		9_1		10_1
$3 \rightarrow 1$	3_2		6_2		7_2				
$4 \rightarrow 1$		4_1							
$5 \rightarrow 2$		5_1		8_1		9_1		10_1	
$6 \rightarrow 3$		6_1							
$7 \rightarrow 3$				7_1					
$8 \rightarrow 5$	8_2								
$9 \rightarrow 5$			9_2						
$10 \rightarrow 5$					10_2				

Figure 7.3: The primary schedule.

Figure 7.4 shows the primary schedule with the bonus slot assignment for the bonus request of node 6 asking for an additional slot. These results are given by the GLPK linear program (in black cells) and by the AMSA algorithm (in grey cells), respectively. In this scenario, we observe that the schedule length is unchanged, despite the bonus path assignment from node 6. This is due to the available spatial reuse.

Slot / $tx \rightarrow rx$	1	2	3	4	5	6	7	8	9
$2 \rightarrow 1$	2_1		5_1		8_1		9_1		10_1
$3 \rightarrow 1$	3_2		6_2		7_2		6'$_2$	6'$_1$	
$4 \rightarrow 1$		4_1							
$5 \rightarrow 2$		5_1		8_1		9_1		10_1	
$6 \rightarrow 3$		6_1				6'$_1$	6'$_1$		
$7 \rightarrow 3$				7_1					
$8 \rightarrow 5$	8_2								
$9 \rightarrow 5$			9_2						
$10 \rightarrow 5$					10_2				

Figure 7.4: The bonus slot assignment for node 6.

The new optimal slot assignment given by MODESA that takes into account the additional demand of node 6 is presented in Figure 7.5. Changes from the primary schedule are represented in grey. Since this schedule is optimal and has the same length as the schedule modified by AMSA, the solution we propose, it follows that there exist scenarios where AMSA provides the optimal solution.

Slot / $tx \rightarrow rx$	1	2	3	4	5	6	7	8	9
$2 \rightarrow 1$	2_1		5_1		8_1		9_1		10_1
$3 \rightarrow 1$	3_2		6_2		6'$_2$		7_2		
$4 \rightarrow 1$		4_1							
$5 \rightarrow 2$		5_1		8_1		9_1		10_1	
$6 \rightarrow 3$		6_1		6'$_1$					
$7 \rightarrow 3$						7_1			
$8 \rightarrow 5$	8_2								
$9 \rightarrow 5$			9_2						
$10 \rightarrow 5$					10_2				

Figure 7.5: The new slot assignment taking into account the additional demand of node 6.

Figure 7.6 describes the bonus slot assignment for the bonus request of node 9 asking for an additional slot. In this case, the bonus slot assignment overflows from the primary schedule, taking slots from the inactivity period (slots 10 and 11).

Slot / $tx \rightarrow rx$	1	2	3	4	5	6	7	8	9	10	11
$2 \rightarrow 1$	2_1		5_1		8_1		9_1		10_1		$9'_1$ $9'_1$
$3 \rightarrow 1$	3_2		6_2		7_2						
$4 \rightarrow 1$		4_1									
$5 \rightarrow 2$		5_1		8_1		9_1		10_1		$9'_1$ $9'_2$	
$6 \rightarrow 3$		6_1									
$7 \rightarrow 3$				7_1							
$8 \rightarrow 5$	8_2										
$9 \rightarrow 5$			9_2				$9'_2$		$9'_2$		
$10 \rightarrow 5$					10_2						

Figure 7.6: The bonus slot assignment for node 9.

With the additional demand of node 9, MODESA obtains the assignment presented in Figure 7.7.

Slot / $tx \rightarrow rx$	1	2	3	4	5	6	7	8	9	10	11
$2 \rightarrow 1$	2_1		5_1		9_1		8_1		$9'_1$		10_1
$3 \rightarrow 1$	3_2		6_2		7_2						
$4 \rightarrow 1$		4_1									
$5 \rightarrow 2$		5_1		9_1		8_1		$9'_1$		10_1	
$6 \rightarrow 3$		6_1									
$7 \rightarrow 3$				7_1							
$8 \rightarrow 5$			8_2								
$9 \rightarrow 5$	9_2				$9'_2$						
$10 \rightarrow 5$							10_2				

Figure 7.7: The new schedule taking into account the additional demand of node 9.

In both examples, AMSA obtains the same schedule length as the linear program; so its bonus slot assignment is optimal for these scenarios. Moreover, the bonus slot assignment has the same number of slots as with a complete re-computation of the optimal assignment obtained with the GLPK model presented in Part II Chapter 5, taking into account the initial and additional transmission needs. We can also deduce from these examples that the length of the new schedule depends not only on the additional bonus slots requested, but also on the depth of requesting nodes in the convergecast tree.

7.4 Performance evaluation

We implemented MODESA and AMSA on a simulation tool based on GNU Octave [Octave] and evaluated the latency (schedule length), energy (radio state switches), bandwidth (slot reuse ratio) and throughput (sink interfaces occupation ratio) in various topologies of WSNs. The number of nodes varies from 10 to 100. We use the Galton-Watson process as a branching stochastic process to generate random trees: each node gives birth to a random number of children independently of the others and according to the same distribution. The number of children is at most equal to three. In addition, we assume that the only existing links are those in the tree. We suppose that all the nodes except the sink have a single radio interface,

and we vary the number of sink radio interfaces from one to three. The maximum number of channels available at each node is equal to two.

We use a comparison with MODESA as a baseline of this study. Indeed, in Chapter 5, we proved that MODESA is optimal for line, multiline and balanced trees. MODESA is the algorithm used to find the primary schedule. We present two different cases of application of our adaptive AMSA algorithm: retransmissions and changing needs of the application. Furthermore, we distinguish two subcases: homogeneous and heterogeneous initial demands of nodes. It is worth noting that in a WSN, we can have different types of wireless sensors running at possibly different sampling rates. This results in non-homogeneous initial demand of nodes. The adaptive scheduling algorithm should be able to handle this heterogeneity.

In addition, in order to show how the convergecast structure can impact the schedule length, we differentiate the two types of configurations, T_t and T_n (see definitions in Chapter 4). This aims at deriving some intrinsic properties of each of them.

In the following, each result is the average of 20 simulation runs for small topologies (≤ 30 nodes) and 100 runs for large topologies.

7.4.1 Retransmission oriented experiments

WSNs are typically deployed in industrial environments, where they are potentially exposed to external interference, making packet losses inevitable, even in a multichannel context. To enhance data reliability and response capability, retransmissions are crucial. In this section, we investigate the behavior of our proposed algorithm AMSA when nodes need to retransmit packets. We evaluate the number of extra slots allocated when each node requests 5%, 10%, 20% of its regular slots as bonus slots. These percentages reflect packet loss probability (low, moderate, high).

7.4.1.1 Homogeneous initial demands

First, we consider the two configurations of 20 nodes illustrated in Figure 7.8(a) and 7.8(b). The first one corresponds to a T_t configuration, whereas the second belongs to the T_n configuration.

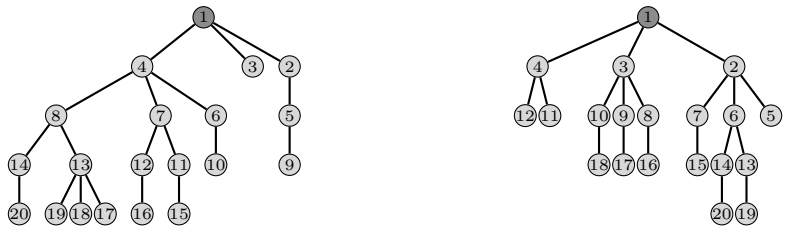
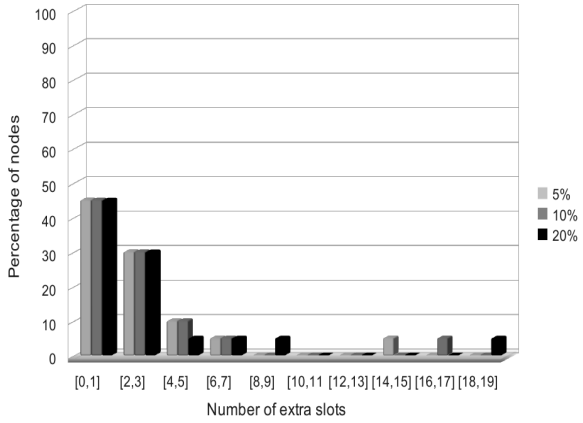


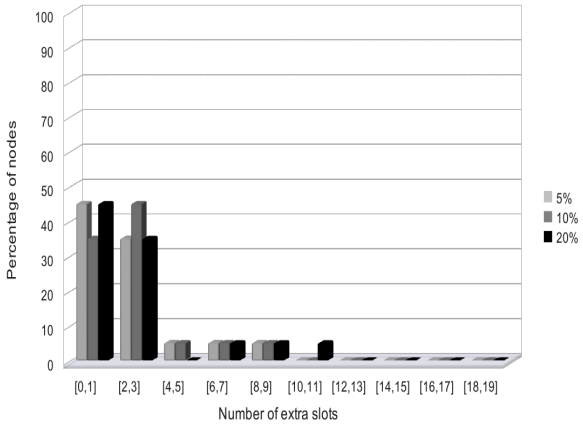
Figure 7.8: Two configurations of 20 nodes. (**a**) T_t configuration; (**b**) T_n configuration.

As mentioned in previous sections, when all nodes initially have one packet to transmit, the number of regular slots for a node is the number of its descendants plus one slot for itself.

The objective is to define which configuration is likely to request more extra slots when the retransmission rate of nodes increases.



(a)



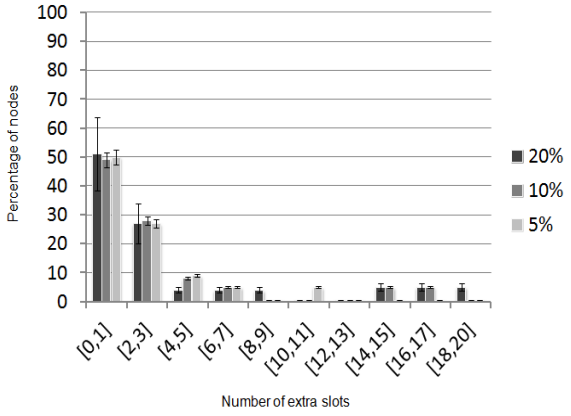
(b)

Figure 7.9: Percentage of required extra slots. (**a**) in T_t configuration 7.8(a); (**b**) in T_n configuration 7.8(b).

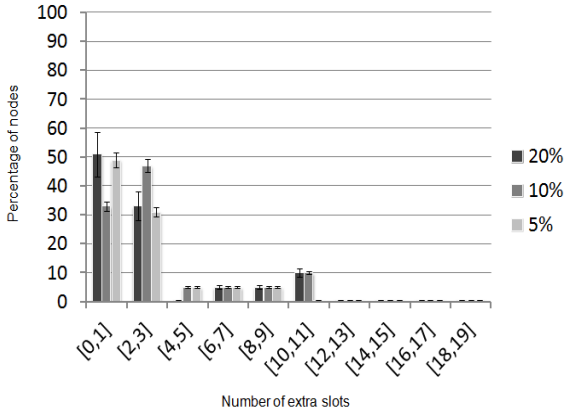
As illustrated in Figure 7.9(a), when the rate of retransmission increases, some nodes in T_t configuration need, respectively, $[14, 15]$, $[16, 17]$ and $[18, 19]$ extra slots for a retransmission rate of 5%, 10% and 20%. Whereas, we notice that for the T_n configuration, the maximum demand for extra slots is less than 12 slots, even when nodes require 20% of their regular slots as bonus slots. Furthermore, we can see that for the T_t configuration, node 4, which is the root of the most populated subtree, is the most greedy in terms of extra slots. For example, for a 5% demand of bonus slots, node 4 requires 15 extra slots in addition to the regular ones. Nevertheless, in the T_n configuration, the number of slots required in addition to the regular ones is balanced between all the nodes. As illustrated in Figure 7.9(b), with a demand increasing up to 10% of the regular slots, 80% of nodes require only zero, one or two or three extra slots. Hence, the impact of bonus requests on the schedule length in the case of retransmission is more drastic in T_t configurations than in T_n configurations. This implies

higher worst case data gathering delays.

To further illustrate this point, we reproduce the same scenario on 25 T_t and 25 T_n configurations of 20 nodes. The previous results are confirmed, as depicted in Figure 7.10: T_t configurations are more greedy in terms of extra slots when the nodes need to retransmit.



(a)

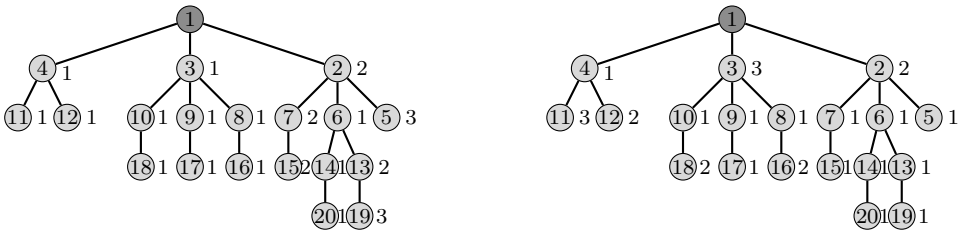


(b)

Figure 7.10: Percentage of required extra slots. (**a**) in T_t configuration; (**b**) in T_n configuration.

7.4.1.2 Heterogeneous initial demands

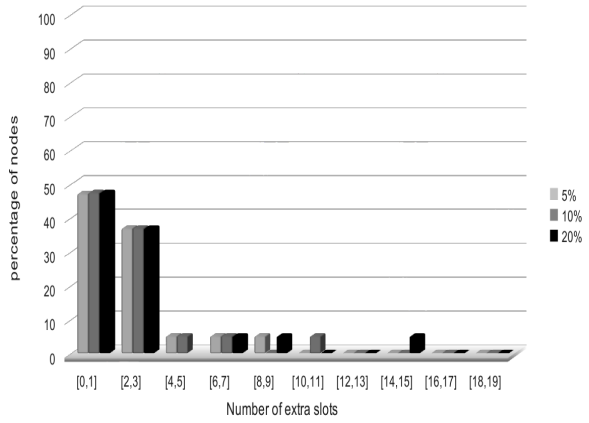
We consider the topology represented in Figure 7.8(b). Notice that this topology can switch from a T_t configuration to a T_n configuration and *vice-versa*, depending on the initial heterogeneous demands. We consider two scenarios of heterogeneous demands. These scenarios have the same global demand of 27, but the demands are distributed differently to get a T_t and a T_n configuration, as depicted, respectively, in Figure 7.11(a) and 7.11(b). The optimal schedule length for the former is 32 slots and 27 for the latter. The two optimal schedules are reached by MODESA.



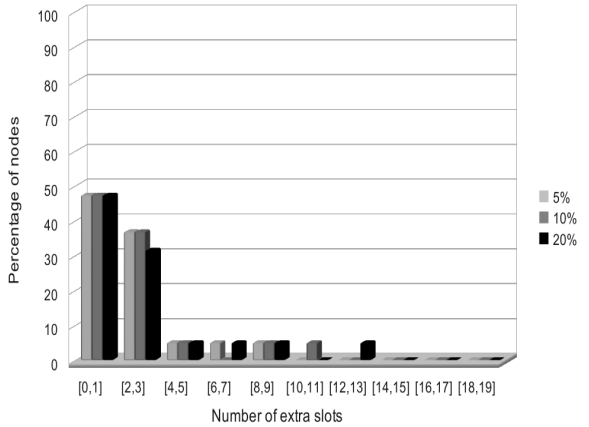
(a) (b)

Figure 7.11: Two configurations for the same topology of 20 nodes. (**a**) T_t configuration; (**b**) T_n configuration.

We conducted the same experiments as in Section 7.4.1.1, varying the retransmission rate of nodes: 5% (low), 10% (moderate) and 20% (high). The results are shown in Figure 7.12(a) and 7.12(b).



(a)



(b)

Figure 7.12: Percentage of extra slots required. (**a**) in T_t configuration; (**b**) in T_n configuration.

Overall, we notice the same conclusions as in the previous section: the T_t configuration needs more extra slots than the T_n configuration.

7.4.2 Temporary change in the application needs-oriented experiments

A WSN can be subject to peaks of traffic. Indeed, an alarm or a physical phenomenon triggers sudden communication activity by nodes. The adaptive scheduling should efficiently handle sudden traffic peaks. This section tackles the problem of changing application demands. In the following, we distinguish two cases: homogeneous and heterogeneous initial demands.

7.4.2.1 Homogeneous initial demands

In the first series of experiments, we consider two configurations of 20 nodes, a T_t configuration and a T_n configuration, as illustrated in Figure 7.8(a) and 7.8(b). We first evaluate the impact on the schedule length of a bonus request made by a node. The node that makes the bonus request is randomly chosen. Figure 7.13 shows the results obtained for each configuration. This experiment enables us to know the distribution of greedy nodes in terms of extra slots.

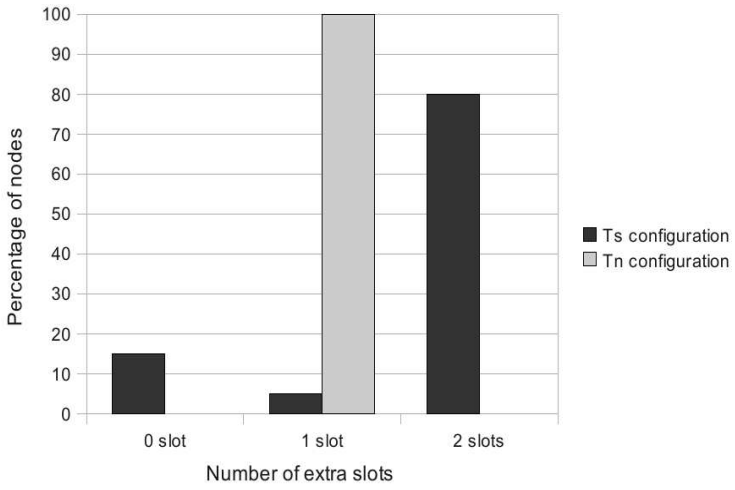


Figure 7.13: Distribution of nodes requiring extra slots.

The results show that for the T_t configuration, 80% of the nodes that retransmit need two extra slots, whereas 5% need only one extra slot and 15% require zero slots. For the T_t configuration, nodes that belong to the dominating subtree need two slots and, therefore, have a higher impact on the schedule length. For example, node 10 in the T_t configuration requires two extra slots, whereas node 9 does not require any extra slot. Moreover, in the T_n configuration, regardless of the node demanding the additional slot, the impact on the schedule length is the same: all nodes need only one extra slot to retransmit the failed packet. This can be explained by the fact that the optimal schedule length is imposed by the number of nodes and their additional demands for slots.

In order to generalize these results, we reproduced the previous experiment with 25 T_t and 25 T_n configurations of 20 nodes. As illustrated in Figure 7.14, we observe that all the

nodes in T_n configurations have the same impact on the schedule length: they add one extra slot. For the T_t configurations, the nodes requiring the highest number of extra slots (two extra slots) belong to the most demanding subtree.

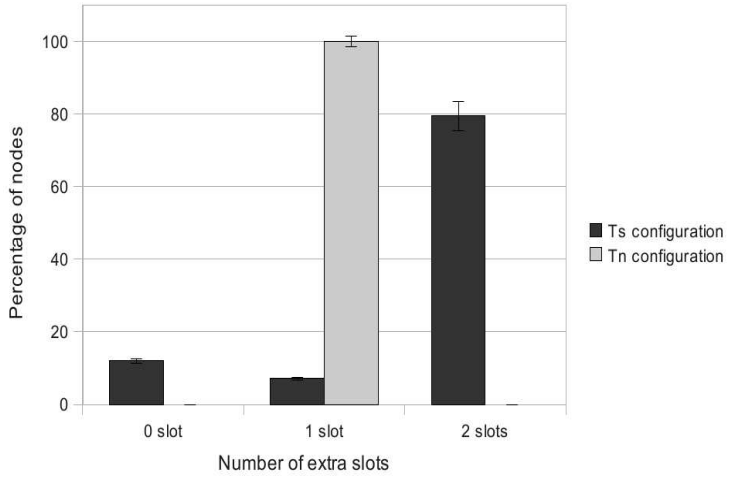
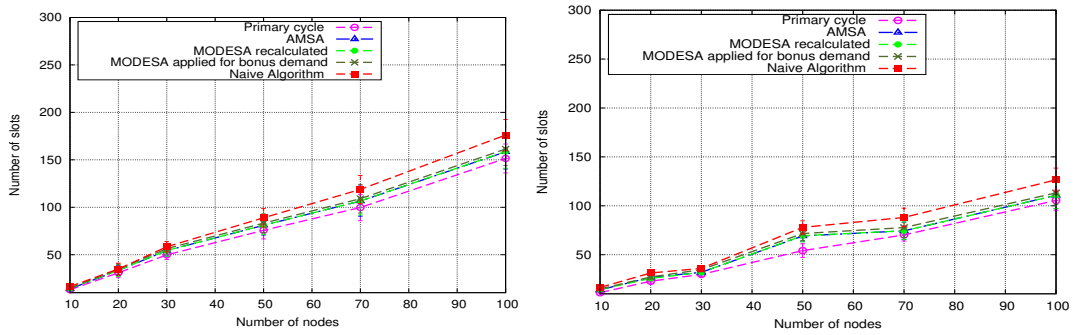


Figure 7.14: Impact on schedule length in case of nodes requiring extra slots.

In the second series of experiments, we compare the performances of AMSA in larger networks with three other approaches:

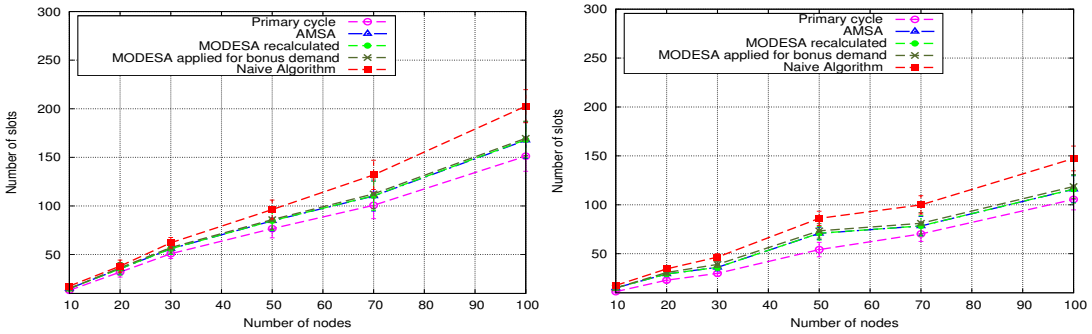
- MODESA recalculated: MODESA is run taking into account both initial and bonus demands.
- MODESA applied for bonus demand: MODESA is run in the primary schedule and also in the complementary schedule to schedule bonus demands. The schedule is formed by the concatenation of the primary schedule and the complementary one.
- Naive Algorithm: assigns each extra slot exclusively to one transmitter. Hence, a bonus demand originated from a node at depth d requires d extra slots.

The primary schedule is given by MODESA, and we vary the percentage of nodes requiring an extra slot (5%, 10% and 20%). We performed the experiments with networks ranging from 10 to 100 nodes. First, we evaluated the average number of extra slots needed with AMSA (*i.e.*, slots created in addition to the primary schedule). The results are depicted in Figures 7.15–7.17 respectively.



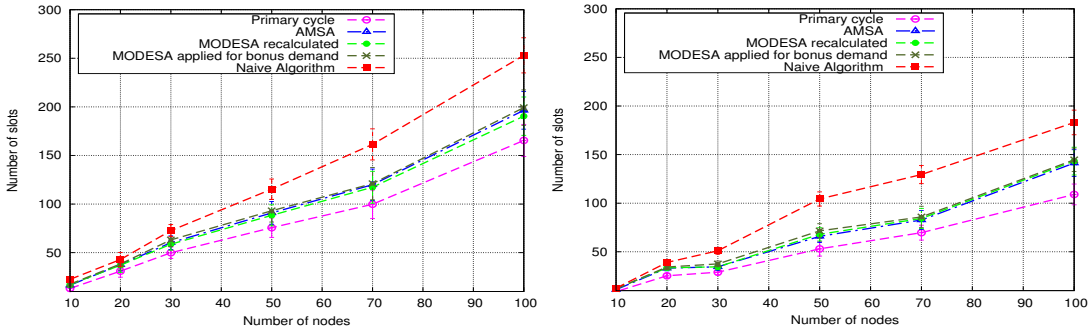
(a) (b)

Figure 7.15: Average number of extra slots required with 5% of nodes having a bonus request. (**a**) in T_t configuration; (**b**) in T_n configurations.



(a) (b)

Figure 7.16: Average number of extra slots required with 10% of nodes having a bonus request. (**a**) in T_t configuration; (**b**) in T_n configurations.



(a) (b)

Figure 7.17: Average number of extra slots required with 20% of nodes having a bonus request. (**a**) in T_t configuration; (**b**) in T_n configurations.

It appears that in both types of configurations (T_t and T_n), the number of extra slots increases with the number of nodes in the topology and the number of nodes requesting bonus slots. This is intuitive, because as the network gets larger, additional packets have to be forwarded to the sink. Note that T_t configurations are more greedy in terms of required extra slots than T_n configurations for the same percentage of nodes having bonus slots.

When 20% of nodes request a bonus slot, MODESA recalculated dominates AMSA more significantly (see T_t configurations in Figure 7.17(a)). AMSA fills the slots in the primary schedule with possible additional transmissions and, then, schedules the remaining ones in extra slots. Nevertheless, MODESA recalculated makes more spatial and frequency reuse by knowing all the demands of nodes at the outset. For example, in T_n configurations, as depicted in Figure 7.17(a), AMSA needs 8 extra slots more than MODESA when 20% of 100 nodes require a bonus slot.

We also notice that MODESA recalculated and AMSA provide very close performances when 5% and 10% of nodes have a bonus request. This illustrates the merit of AMSA that provides the same performance level as MODESA recalculated, but with less complexity.

Figure 7.17 also shows that MODESA applied for bonus demands requires more slots to schedule additional demand than AMSA and MODESA recalculated. This can be explained by the fact that MODESA applied for bonus demands computes the complementary schedule separately from the primary one. Only bonus demands are scheduled in this complementary schedule. Thus, its schedule is not as optimized as MODESA recalculated.

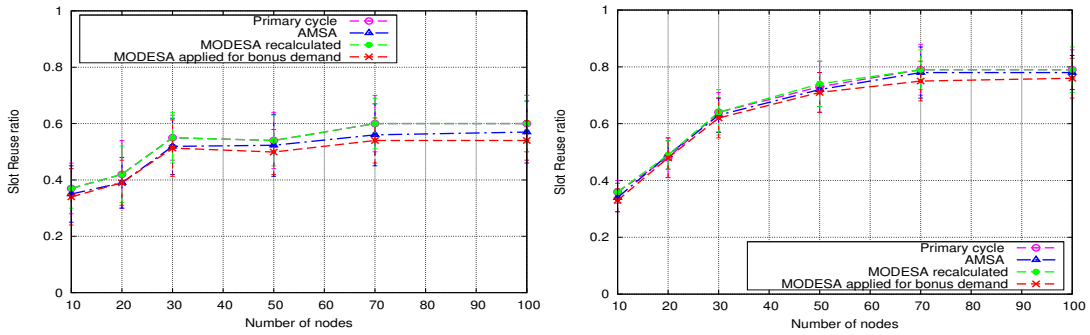
This second series of experiments allows us to make some conclusions regarding the efficiency of AMSA that takes advantage of spatial reuse and multichannel to assign bonus slots.

In the third series of experiments, we varied the number of nodes from 10 to 100 and assumed that 20% of nodes request a bonus slot. We compared the AMSA assignment with the three other approaches (MODESA recalculated, MODESA applied for bonus demand, primary schedule) in terms of energy (evaluation of radio state switches), bandwidth (evaluation of slot reuse) and throughput (evaluation of sink interfaces occupation ratio). All the results are presented in Figures 7.18–7.20.

Overall, the performance of MODESA recalculated and AMSA are very close. More specifically, in T_n configurations, MODESA recalculated has slightly better performance than AMSA. For example, in Figure 7.18(b), the two approaches achieve the same slot reuse ratio. However, in T_t configurations, MODESA recalculated slightly outperforms AMSA, as shown in Figure 7.18(a), achieving higher slot reuse ratio. MODESA applied for bonus demands achieves a lower slot reuse than the other two approaches. Undoubtedly, the slot reuse ratio significantly reduces the end-to-end latency without any adverse effect on energy efficiency.

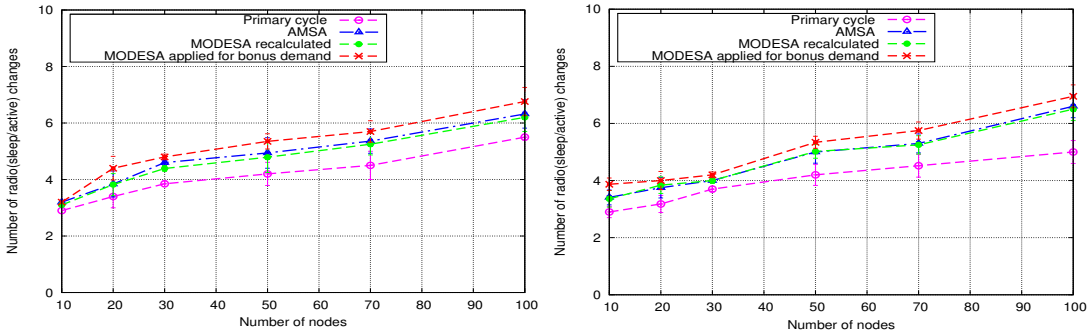
Similarly, as depicted in Figure 7.19(a) and 7.19(b), the number of radio state switches of MODESA recalculated matches that of AMSA. This number of switches that takes into account both initial and bonus demands is not far from that achieved in the primary schedule. Nevertheless, MODESA applied for bonus demands leads to more radio state switches. Thus, both AMSA and MODESA decrease energy consumption, due to state switches.

We also explored the throughput by means of the sink radio interface occupation ratio. This is defined as the number of slots during which the sink receives packets divided by the



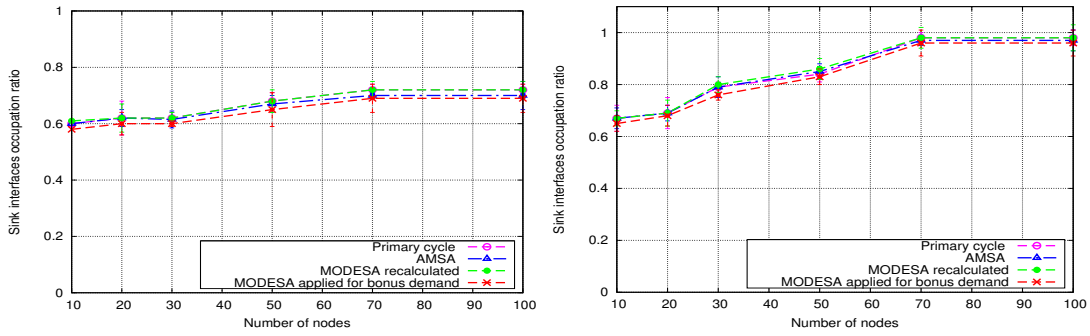
(a) (b)

Figure 7.18: Slot reuse ratio with 20% of nodes having a bonus request. (**a**) in T_t configuration; (**b**) in T_n configurations.



(a) (b)

Figure 7.19: Average number of state switches per node with 20% of nodes having a bonus request. (**a**) in T_t configuration; (**b**) in T_n configuration.



(a) (b)

Figure 7.20: Sink interfaces occupation ratio with 20% of nodes having a bonus request. (**a**) in T_t configuration; (**b**) in T_n configuration.

total number of slots. Figure 7.20 illustrates that MODESA recalculated guarantees a slightly higher throughput than AMSA. Another remarkable phenomenon is that T_n configurations outperform T_t configurations in terms of throughput. Indeed, in T_t configurations, the sink interface is kept busy only one slot over two until the dominating subtrees have completed their transmissions. This property is true whatever the number of sink interfaces and the number of channels.

7.4.2.2 Heterogeneous initial demands

We again consider the two types of convergecast configurations: T_t and T_n, as depicted, respectively, in Figure 7.11(a) and 7.11(b). In these configurations, the initial demands of nodes are heterogeneous. The sink has a single interface, and nodes can transmit on three channels. As in Subsection 5.1.1, we first evaluate the impact on the schedule length of a bonus request made by a node. The node that makes the bonus request is randomly chosen. Figure 7.21 shows the results obtained in each configuration.

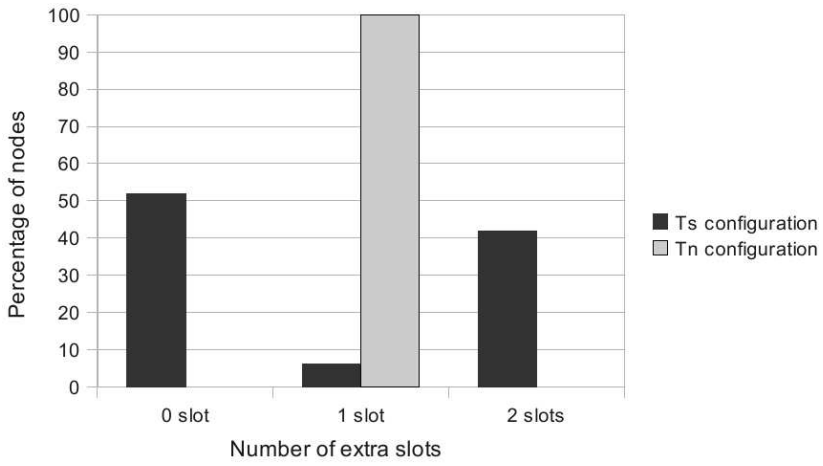


Figure 7.21: Distribution of nodes requiring extra slots.

We find the same conclusions as in the case of homogeneous initial demand: for the T_t configuration, 43% of the nodes that retransmit need two extra slots, while 5% need only one extra slot and 52% require zero slots. Nodes that need two extra slots belong to the dominating subtree. Moreover, for the T_n configuration, we come to the same conclusion: regardless of the node demanding the additional slot, the impact on frame length is the same: all nodes need one extra slot to retransmit the failed packet. All the nodes have the same impact on schedule length.

In order to generalize these results, we reproduced the previous experiment with 25 T_t and 25 T_n configurations of 20 nodes. Initially, nodes have heterogeneous demands between one and five. As shown in Figure 7.22, we observe that all the nodes in T_n configurations have the same impact on the schedule length: they add one extra slot. For the T_t configurations, the nodes requiring the highest number of extra slots (two extra slots in our simulation scenario) belong to the most demanding subtree.

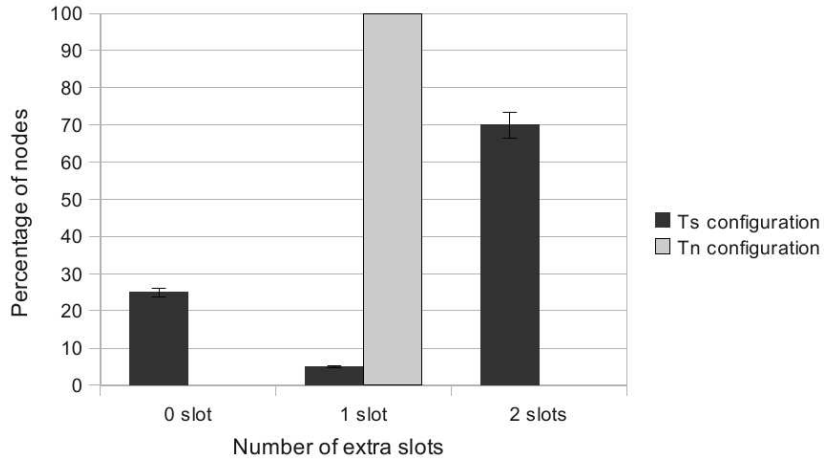
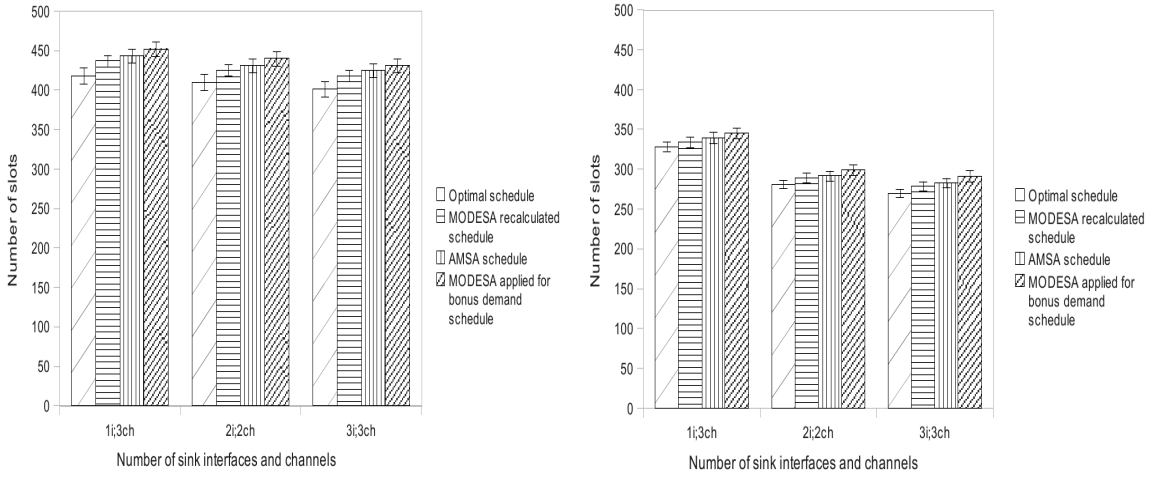


Figure 7.22: Impact of demands for bonus slots on the schedule length.

After that, we considered larger topologies with 100 nodes. We varied the number of sink radio interfaces, as well as the number of channels. As mentioned previously, we compared the three approaches (AMSA, MODESA recalculated and MODESA applied for bonus slots) in terms of the number of slots. In this group of simulations, all the nodes initially have heterogeneous demands, uniformly distributed between one and five. In addition, 20% of the nodes request a bonus slot.



(a) (b)

Figure 7.23: Number of slots required for convergecast in (**a**) T_S configurations (**b**) T_N configurations.

The simulation results show, as depicted in Figure 7.23, that T_t configurations are more greedy in terms of slots needed for raw data convergecast than T_n configurations. The T_n configurations are more balanced in terms of nodes demands per subtree rooted at a sink child. Hence, their schedule tends to be shorter. Consequently, we can make some recommendation

regarding the routing tree built: **the routing tree built should balance the nodes demands on subtrees rooted at a sink child**. These results generalize the work of Incel *et al.* [Incel 2012] when nodes have heterogeneous demands.

We have also compared the number of slots obtained by AMSA and MODESA recalculated with the optimal number of slots. For a sink with one radio interface and three channels, MODESA recalculated (respectively AMSA) is optimal in 50% of the T_t configurations (respectively, 43.18%) and in 87% of the T_n configurations (respectively, 82%), as illustrated in Figure 7.24.

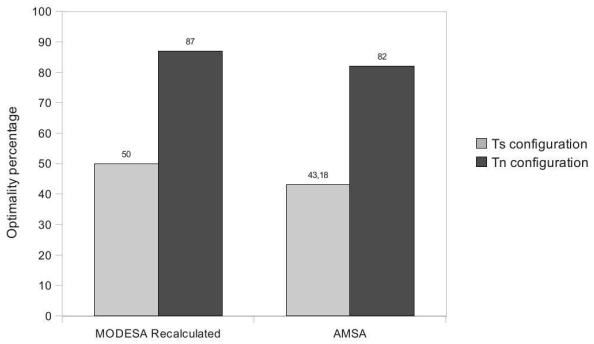


Figure 7.24: Optimality of MODESA recalculated and AMSA in T_t and T_n configurations.

In addition, we evaluate the distance of AMSA and MODESA recalculated to the optimal schedule in configurations where they are not optimal. This distance is called inaccuracy and is computed as: $\frac{number\ of\ slots\ needed - optimal\ number\ of\ slots}{optimal\ number\ of\ slots}$. As depicted in Figure 7.25, the average inaccuracy of AMSA is 9.2% for the T_t configurations and 3.2% for the T_n configurations, demonstrating the very good behavior of AMSA. The average inaccuracy in both T_t and T_n configurations is below 10%.

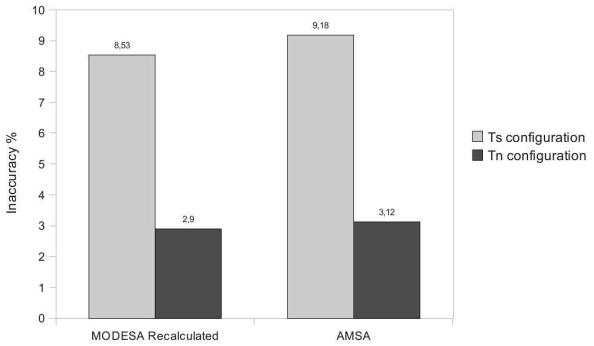


Figure 7.25: Inaccuracy of MODESA recalculated and AMSA in T_t and T_n configurations.

These experiments demonstrate that AMSA and MODESA recalculated have comparable performances in terms of extra slots. This means that AMSA performs well in filling existing slots with possible transmissions without the need to recompute the schedule again unlike MODESA recalculated. The light weight of AMSA considerably enhances its adaptivity in networks with different traffic loads and additional demands. In addition, using multi-radio

interfaces for the sink with multiple channels decreases the frame length. Hence, the activity period of nodes is decreased, allowing them to sleep more to save their limited energy.

7.5 Conclusion

The unexpected traffic loads or retransmissions can hamper the initial time slot allocation for data convergecast in a WSN. In order to address this issue, we tackled the problem of adaptive slot assignment in multichannel WSNs. We have formalized and solved this problem with linear programming. We have shown that in T_t configurations, where the most demanding subtree imposes the length of the primary schedule, the request for bonus slots made by nodes belonging to this subtree has a great impact on the number of extra slots required, unlike in T_n configurations, where the number of extra slots is much more balanced between all the nodes. As a conclusion of this study, we recommend the use of a routing tree that balances the demands of nodes among the subtrees.

We then proposed the AMSA algorithm, which unifies the management of additional slots, whatever their origin: changes in the application demands or in the medium access demands due to retransmissions. AMSA assigns bonus slots in an optimized primary schedule provided by MODESA, adding extra slots only when the slots allocated in the primary schedule are unable to meet the dynamic demands. It takes advantage of spatial reuse to reduce the number of bonus slots added to the regular ones. We evaluated the performances of AMSA in various multichannel topologies and how close it comes to an optimized schedule. The adaptivity of AMSA to both changes in the application or in the medium access demands ensures efficient convergecast in WSNs, especially in the case of low (5%) and moderate (10%) additional transmissions. When 20% of nodes require additional slots, AMSA is optimal in 43% of the T_t configurations and 82% in the T_n configurations. In all cases, AMSA's schedule length is less than 10% higher than the optimal one.

Nevertheless, centralized algorithms are inherently inefficient in large scale WSNs. That is why, the next chapter will tackle scalability, the last issue in Part III.

Chapter 8

A Distributed Joint Channel and Time Slot Assignment for Convergecast in Multichannel WSNs

Contents

8.1 Introduction . . . 135
8.2 State of the art . . . 136
8.3 WAVE: a distributed time slot and channel assignment for convergecast in WSNs . . . 137
8.3.1 WAVE in centralized mode . . . 137
8.3.2 WAVE in distributed mode . . . 142
8.3.3 Properties . . . 145
8.3.4 Equivalence of the two modes of WAVE . . . 146
8.3.5 Message complexity in the two modes of WAVE . . . 146
8.4 DiSCA: an optimized version of WAVE . . . 148
8.4.1 Principles . . . 149
8.4.2 DiSCA distributed algorithm . . . 150
8.5 Comparative performance evaluation of WAVE and DiSCA . . . 153
8.5.1 Homogeneous traffic . . . 153
8.5.2 Heterogeneous traffic . . . 155
8.6 Conclusion . . . 156

8.1 Introduction

The emerging of data intensive applications in WSNs requires solutions for high data rate convergecast. Distributing the communication across multiple channels for parallel transmissions is an attractive solution. In addition, as the size of the WSN grows, centralized

scheduling is not scalable and less adaptive to network changes.
Moreover, as convergecast involves a large number of sensors that may transmit simultaneously, collisions and retransmissions represent a major challenge for bounded latencies. Indeed, collisions lead to data losses. Retransmissions increase packet latency and result on non-deterministic packet delivery times. Unlike contention-based protocols which suffer from inefficiency due to backoff and collisions, collision-free protocols guarantee bounded latencies. It is achieved by allocation of channels and time slots to nodes in such a way that these interferences are avoided. Thus, it is easy to control the packet delay needed to reach the final destination.

Recently, the MAC amendment, *IEEE* 802.15.4*e* TSCH [TG4e 2012] adds functionality to better meet industrial markets requirements. The TimeSlotted Channel Hopping (TSCH) mode ensures robustness and high reliability against interferences by channel hopping. *IEEE* 802.15.4*e* highlights how the MAC layer executes the nodes schedule. Nevertheless, the policy that provides such a schedule is not specified.

That is why in this chapter, we propose *WAVE*, a simple and interference-aware distributed joint channel and time slot assignment for a traffic-aware raw data convergecast. Unlike most previous studies, we consider two cases of transmissions: without acknowledgment and with immediate acknowledgment. Furthermore, we prove that WAVE in distributed mode is equivalent to its centralized mode. We evaluate the number of slots needed to complete the convergecast by simulation and compare it to the optimal schedule and to our centralized solution MODESA. Furthermore, we design an optimized version of *WAVE*, called DiSCA that is closer to the optimal as we will see with the simulations results. Besides, this improvement comes at the price of a higher complexity.

8.2 State of the art

We will only focus on distributed approaches for channel and slot assignment in WSNs. Such approaches are considered to be more scalable and reliable than centralized ones.

Authors of [Lin 2011] tackle applications with high traffic load and minimized delays on single channel WSNs. They propose GLASS, a distributed conflict-free schedule access. GLASS operates in three phases. First, each sensor associates itself to one virtual grid. Second, sensors located in adjacent cells use two orthogonal subtransmissions frames. Finally, the Latin square matrices are used to assign time slots to sensors within a grid cell. The CAIG mechanism avoids collisions in the intersection of adjacent grid cells. GLASS is not oriented toward raw data convergecast: only a single slot is granted per node. This is usually not sufficient in the absence of aggregation.

Recently, many studies have resorted to multichannel communications in order to deliver collected data in a short time. In [Durmaz Incel 2008], Incel et al. derive a TDMA schedule that minimizes the number of slots required for convergecast. They prove that a lower bound for raw convergecast is $max(2 * n_k - 1, N)$ where n_k is the maximum number of nodes on any subtree and N is the number of nodes in the network. They extend the distributed algorithm proposed by Ghandam et al. [Gandham 2008] to the context of multichannel WSNs. Their approach includes two steps:

(1) a receiver based channel assignment: it removes all the interference links in an arbitrary network.
((2) a distributed slot assignment: where each node is assigned an initial state (i.e. transmit T_x, receive R_x or idle) based on its hop-count to the sink and the state of its branch. If the branch is active (i.e. the sink child located in the top of the branch transmits), a node with hop-count h is assigned state T_x if $h \ mod \ 2 = 1$ and state R_x otherwise. If the branch is not active, it is assigned state T_x if $h \ mod \ 2 = 0$ and R_x otherwise. In the next slot, nodes switch to the opposite state.
To eliminate conflict between two children of the same node, the algorithm assumes that any node should know the number of remaining packets for its brothers to schedule them in round robin order. Besides, the algorithm assumes that the only remaining conflicts are inside the convergecast tree after the completion of the first phase.

In [Han 2009], a distributed joint channel allocation of links and packet scheduling for Software-Defined Radio, called CLDS (Collision-free Distributed Scheduling) combines the use of an access hash function with the inductive scheduling technique. The hash function SHA-1 allows CLDS to know if a pair of nodes will communicate or not on a channel and in the current time slot. In addition, to be collision-free, a node needs to exchange its links utilization with its interfering nodes. However, CLDS is not designed for convergecast applications. Hence, a slot can be granted to a node even if this node has no packet to transmit. Furthermore, the end-to-end delays can be large due to an assignment of slots that does not take into account the progression of data toward the sink.

Authors of [Accettura 2013] propose DeTAS, a distributed traffic aware scheduling solution for $IEEE$ 802.15.4e $TSCH$ networks. This solution is the distributed mode of TASA proposed in [Palattella 2012]. In DeTAS, all nodes follow a common schedule, called macro-schedule, that is the combination of micro-schedules of each routing graph. This solution optimizes the number of slots needed by the data gathering. However, if other links exist in addition to the convergecast links, collisions may occur.

8.3 WAVE: a distributed time slot and channel assignment for convergecast in WSNs

In this section, we propose $WAVE$, a joint slot and channel assignment algorithm. We give the centralized and distributed modes of $WAVE$ and prove their equivalence.

8.3.1 WAVE in centralized mode

The $WAVE$ algorithm schedules nodes in successive waves. In each wave, each node having a packet to transmit is assigned a time slot and a channel. The first wave constitutes the (slot x channel) pattern. Each next wave is an optimized subset of the first wave: only the slots that will contain transmissions are repeated and they always occur in the same order as in the first wave. As a result, the joint channel and time slot assignment produced by $WAVE$ contains for each time slot and for each available channel, a list of sender nodes, such that their transmissions to their parent do not conflict.

8.3.1.1 Principles

More precisely, the $WAVE$ algorithm proceeds as follows:

- *Building of the first wave*: Each node u has a priority equal to $Trans(u)$ the number of slots needed to transmit its own data and the data received from its children. $WAVE$ schedules nodes in the order of their decreasing priority (see Algorithm 6), granting them the smallest available time slot and the first available channel (channels being visited according to a round-robin strategy, see Algorithm 5). Let P be the (slot x channel) pattern obtained and T be the number of time slots in P.
- *Computation of the next waves*: $WAVE$ repeats the pattern P of the first wave a number of times equal to W where W is equal to the maximum number of transmissions done by a node in a data gathering cycle: $W = max(Trans(u))$. However, the repeated pattern is optimized according to the following rule:

Rule R0: Any node u having slot j and channel c in the first wave, with $1 \leq j \leq t$, has also the slot $s(k)$ and channel c in any wave k with $1 \leq k \leq Trans(u)$,

$$s(k) = \sum_{w=1}^{k-1} \sum_{t=1}^{T} \delta_{t,w} + \sum_{t=1}^{j} \delta_{t,k} \tag{8.1}$$

$\delta_{t,w} = 1$ if and only if $MaxTrans(t) \geq w$ and 0 otherwise. $Maxtrans(t)$ is the maximum number of transmissions of any node transmitting in the slot t, T is the number of slots in the pattern produced by the first wave.

The $WAVE$ algorithm produces W successive waves. In each wave, each node having not acquired all its slots requested, is granted exactly one time slot. The assignment produced by $WAVE$ contains exactly $\sum_{1 \leq t \leq T} Maxtrans(t)$ slots.

Let us consider the example depicted in Figure 8.1 with a pattern of 5 slots, $P = A\ B\ C\ D\ E$ such that $Maxtrans(A) = 6$, $Maxtrans(B) = 5$, $Maxtrans(C) = 3$, $Maxtrans(D) = 2$ and $Maxtrans(E) = 1$. In this figure, the number inside the slot indicates the maximum remaining transmissions of a node scheduled in this slot at the beginning of the i^{th} wave. The scheduling provided by the $WAVE$ algorithm contains 6 waves and 17 slots that are $A\ B\ C\ D\ E$, $A\ B\ C\ D$, $A\ B\ C$, $A\ B$, $A\ B$, A.

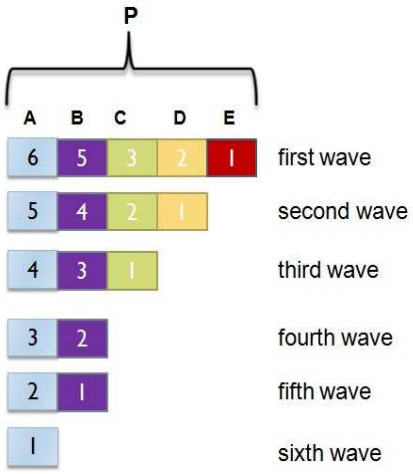


Figure 8.1: Illustrative example for the $WAVE$ algorithm

Notice that for any node u, its conflicting nodes $Conflict(u)$ are an input of the $WAVE$ algorithm in both modes. The determination of this set depends on the policy used for the acknowledgment (immediate acknowledgment or no acknowledgment). Hence, the same algorithm is applied whatever the policy used for the acknowledgment, but with different inputs. The algorithm is given hereafter.

8.3.1.2 WAVE centralized algorithm

The pseudocode of centralized mode of WAVE is depicted in Algorithm 6. This latter fixes all nodes schedules. To search for suitable time slot and channel to schedule a node, $WAVE$ resorts to the $ScheduleNode$ function detailed in Algorithm 5.

Algorithm 5 ScheduleNode Function (v, $Conflict(v)$, $nchannel$, $ScheduledNodes$)

Input: node v, $Conflict(v)$: the set of conflicting nodes, $nchannel$: the number of channels, $ScheduledNodes$: the table of scheduled nodes per slot and channel.
Output:
- $slot$: the slot assigned to node v
- c: channel allocated to node v
- $ScheduledNodes$: scheduled nodes per slot and channel.

Initialization:
$c \leftarrow 1$
$tx \leftarrow false$
$nChannelReached \leftarrow false$
$t \leftarrow 0$
while $tx == false$ **do**
  while $(AvailInter(v,t) = 0 \;||\; AvailInter(Parent(v),t) = 0\;)$ **do**
    /*find a time slot with available interface for v & $Parent(v)$*/
    $t \leftarrow t + 1$
  end while
  repeat
    if $(Conflict(v) \bigcap ScheduledNodes(t,c) = \emptyset)$ **then**
      /*Node v can be scheduled */
      $tx \leftarrow true$;
      Node v transmits in slot t on channel c
      $AvailInter(v,t) \leftarrow AvailInter(v,t) - 1$
      $AvailInter(Parent(v),t) \leftarrow AvailInter(Parent(v),t) - 1$
      $ScheduledNodes(t,c) \leftarrow ScheduledNodes(t,c) \cup \{v\}$
    else
      if $(c < nchannel)$ **then**
        $c \leftarrow c + 1$ // try the next channel
      else
        $nChannelReached \leftarrow true$
      end if
    end if
  until $(tx \;||\; nChannelReached\;)$
  if $(tx == false)$ **then**
    $t \leftarrow t + 1$ // no channel available for v in slot t
  end if
end while
return (t, c, $ScheduledNodes$)

Algorithm 6 WAVE algorithm: first wave in centralized mode

Input: $nchannel$ channels; a routing tree T where each node u has a set of conflicting nodes $Conflict(u)$, a number of available radio interfaces $AvailInter(u)$ and a priority $Trans(u)$=number of packets that node u should transmit.
Output: $ScheduledNodes$: Channel and time slot assignment for all nodes
Initialization:
$scheduledNodes \leftarrow \emptyset$
Channel and slot assignment for all nodes:
$SortedNodesList \leftarrow$ nodes sorted by decreasing priorities
$u \leftarrow first(SortedNodesList)$
while $SortedNodesList \neq \emptyset$ **do** // there are nodes to schedule
  ($slot$,c,$ScheduledNodes$)=ScheduleNode(u,$Conflict(u)$, $nchannel$,$ScheduledNodes$)
  remove(u) from $SortedNodesList$
  $u \leftarrow next(u)$
end while // no pending demand

8.3.2 WAVE in distributed mode

We now focus on the distributed mode of $WAVE$.

8.3.2.1 Principles

The main difference between the two modes concerns the knowledge required. In centralized mode, the central entity running $WAVE$ has the knowledge of any information related to each node. In distributed mode, any node u knows only the information related to its set of conflicting nodes. To acquire this information, messages are exchanged between neighboring nodes. Furthermore, a specific message, called *SlotChAssigned*, is created to notify the neighboring nodes of a (time slot, channel) assignment. The processing and forwarding algorithms of this message are given in Algorithm 7 and 8 respectively.

In distributed mode, the global time slot and channel assignment is built by assembling all the partial assignments known by nodes. More practically, to compute the first wave as shown in Algorithm 9, it is sufficient for each node u to know: its conflicting nodes $Conflict(u)$ and the number of transmissions $Trans(v)$ of each node $v \in Conflict(u)$.

Then any node u sends to its parent the maximum number of transmissions for each slot it knows. At the end of the first wave, the sink computes T the total number of slots in the pattern and for each slot t with $1 \leq t \leq T$, its maximum number of transmissions $Maxtrans(t)$. It sends this information to all nodes via the routing tree. Nodes are now able to compute the next waves according to Rule $R0$.

More precisely, $WAVE$ in distributed mode applies the following rules:
Rule R1: For any node $u \neq sink$, $WAVE$ defines the priority of any node as follows: $Prio(u) = \sum_{v \in Subtree(u)} Gen(u)$.
Rule R2: $WAVE$ in distributed mode proceeds in scheduling waves as illustrated in Algorithm 9: on any node u, the i^{th} wave schedules the i^{th} transmission of any node $v \in Conflict(u)$ with unscheduled transmissions. More precisely, the i^{th} transmission of any node $u \neq sink$ with $Trans(u) \geq i \geq 1$ is scheduled in wave i if and only if:

- the i^{th} transmission of any node $v \in Conflict(u)$ such that $Prio(v) > Prio(u)$, with $1 \leq i \leq Trans(v)$ is already scheduled,
- and if $i > 1$, the $(i-1)^{th}$ transmission of any node $v \in Conflict(u)$, whatever its priority, with $1 \leq i-1 \leq Trans(v)$, is already scheduled.

Rule R3: each node u whose i^{th} transmission is scheduled in slot t and on channel c notifies its conflicting nodes with the *SlotChAssigned* message (see Algorithm 8).

Lemma 8. *Any node v whose transmission is scheduled, sends a SlotChAssigned message to its 1-hop neighbors. Any node u receiving such a message, forwards it if and only if:*
- Case 1 no acknowledgment: u has children and is 1-hop away from v.
- Case 2 immediate acknowledgment: u has children and is 1-hop away from v or the parent of v.

Proof. Let us assume that the transmission of v is scheduled on channel ch in time slot t. Hence, v knows this assignment (case a from Lemma 1 Chapter 4). We distinguish two cases:

◦ *Without acknowledgement* :
- Since v sends the *SlotChAssigned* message, any node 1-hop away from v knows this assignment, in particular the parent of v and the children of v.
- Since $Parent(v)$, 1-hop away from v, has children, it forwards; hence all nodes 1-hop away from $Parent(v)$ (case 1 Figure 8.2) know the assignment .
- Since all nodes 1-hop away from v and having children forward, their children know the assignment (case 2 Figure 8.2). Hence, all nodes in Lemma 1 Chapter 4 know the assignment.

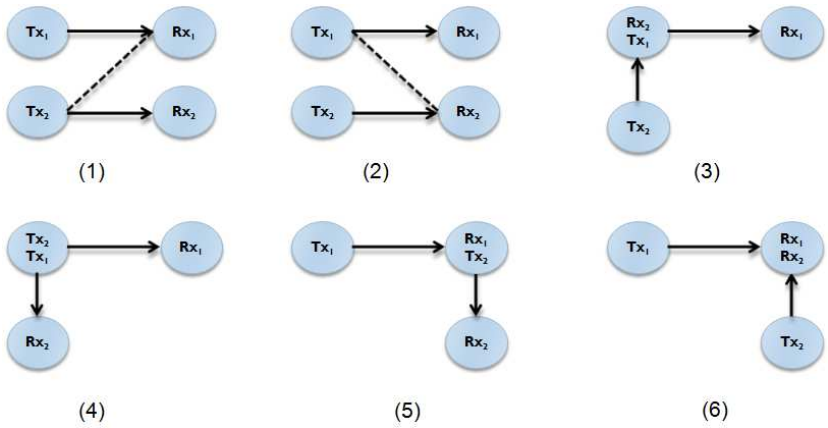


Figure 8.2: Conflicting transmissions without acknowledgment.

◦ *With immediate acknowledgement* :
- Proof idem for cases a and b
- Since v sends its message, all nodes 1-hop away from v know the channel and slot assignment (case 8 Figure 8.3).
- Since $Parent(v)$ forwards the message, all nodes that are 1-hop away from $Parent(v)$ know this assignment (case 8 Figure 8.3).
- Since all nodes, 1-hop away from v, having children forward, their children know the assignment (case 7 Figure 8.3).
- Since all nodes, 1-hop away from $Parent(v)$ having children forward, their children know the assignment (case 7 Figure 8.3). Hence, all nodes listed in Lemma 2 know the assignment.

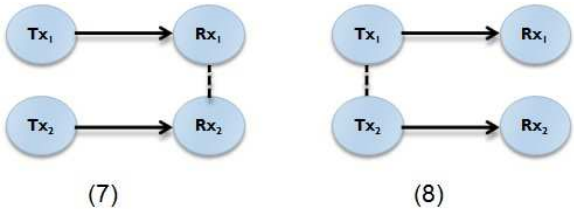


Figure 8.3: Additional conflicting transmissions with immediate acknowledgment.

□

Rule R4: **the i^{th} transmission of u is scheduled in the first slot t after the $(i-1)^th$ wave** and on the first channel c (channels being visited in round robin) where the following conditions are met:

- u has a packet to transmit,
- u has an available interface,
- the parent of u has an available interface,
- the transmission of u does not conflict with the transmission of any node already scheduled in the slot t and on the channel c.

8.3.2.2 WAVE distributed algorithm

The pseudocode of WAVE is depicted in Algorithm 9. This latter fixes a node u's schedule in the first wave. When u schedules its transmission in slot t and channel c, it notifies its conflicting nodes with the *SlotChAssigned* message as illustrated in Algorithm 8. Moreover, Algorithm 7 defines the needed actions when a node u receives *SlotChAssigned* from another node v.

Algorithm 7 processRecSlotChAssigned Procedure (*SlotChAssigned*)

```
Input: message SlotChAssigned(slot, c, v, Parent(v))
if v ∈ Conflict(u) then
    update ScheduledNodes
    Slot(v) ← 1
else if v = Child(u) then
    update AvailInterf(u, slot)
else if (v = Parent(u)) || (Parent(v) = Parent(u) & v ≠ u) then
    update AvailInterf(Parent(u), slot)
end if
```

Algorithm 8 forwardRecSlotChAssigned Procedure (*SlotChAssigned*)

```
Input: message SlotChAssigned(slot, ch, v, Parent(v))
if Ack not enabled then
    if (u has child) & (u is 1-hop away from v) then
        forward SlotChAssigned
    end if
else /* immediate acknowledgment */
    if (u has child) & (u is 1-hop away from v or Parent(v)) then
        forward SlotChAssigned
    end if
end if
```

Algorithm 9 WAVE algorithm: first wave in distributed mode

Input: $nchannel$: the number of channels; T: the routing tree where the local node u has a set of conflicting nodes $Conflict(u)$, $AvailInter(u)$: a number of available radio interfaces and a priority $Trans(u)$=number of packets that node u should transmit.
Output: $ScheduledNodes$: Channel and time slot assignment for local node u and $Conflict(u)$ per slot and channel
Initialization:
$\forall v \in Conflict(u)$, $Slot(v) \leftarrow 0$ /*number of slots already assigned to node u*/
Channel and slot assignment for node u
$SortedNodesList(u) \leftarrow$ the set $\{u\} \cup Conflict(u)$ sorted by decreasing priorities.
while $\exists v \in Conflcit(u) such that Slot(v) = 0$ **do**
  if reception of a $SlotChAssigned(slot, ch, v, Parent(v))$ Message **then**
    $processRecSlotChAssigned()$ /* see Algorithm 7
    $forwardRecSlotChAssigned()$ /*forward the received information according to Algorithm 8
  end if
  if ($Slot(u) = 0$) & ($\forall v \in SortedNodesList(u)$ such that $Trans(v) > Trans(u)$, where $Slot(v) > Slot(u)$) **then**
    ($slot$,ch,$ScheduledNodes$)=ScheduleNode(u,$Conflict(u)$, $nchannel$, $ScheduledNodes$)
    $Slot(u) \leftarrow 1$
    transmission of the $SlotChAssigned(slot, ch, u, Parent(u))$ Message to the 1-hop neighbors of u.
  end if
end while

8.3.3 Properties

Property 4. *Efficiency: WAVE, under the assumption given in Chapter 4 Section 4.2 ensures that:*

- *no slot is empty;*
- *if the transmission of a node is scheduled in a given slot, this node has a message to transmit;*
- *if a packet is transmitted in a data gathering cycle, it reaches the sink in this cycle.*

Proof. According to the principles of the algorithm, only nodes having at least one packet to transmit are scheduled in the current wave. Furthermore, the number of remaining packets of any node at the beginning of a wave is such that it reaches 0 only when this node has completed all the transmissions required by the convergecast. We deduce that no slot is empty. Furthermore, in each wave, the transmitted messages progress toward the sink. The number of waves W is computed to allow any transmitted message to reach the sink. Hence the property. □

Property 5. *Simplicity and adaptivity: WAVE easily adapts to traffic changes.*

Proof. When the traffic changes, the first wave is kept. Only the computation of the next waves is redone, preserving the initial pattern while taking into account the new values of $Trans(u)$ for any node $u \neq sink$. Hence the property. □

8.3.4 Equivalence of the two modes of WAVE

Under the assumptions 1, 2 and 3 given in Chapter 4, the $WAVE$ algorithm exhibits the following property:

Property 6. *Equivalence of centralized and distributed modes: For any considered topology of WSN, for any raw data convergecast and for any data traffic, the distributed and the centralized modes of $WAVE$ provide the same slot and channel assignment.*

Proof. First, we prove that both modes of $WAVE$ produce the same first wave. All nodes $\neq$ sink have at least one packet to transmit. Hence, they are all scheduled in the first wave in both modes. We have only to prove the equality of the assignments produced. We proceed by contradiction. Let t be the first slot where the assignments differ: there is a node v that is scheduled in slot t and on channel c by one mode and in slot t' and on channel c' by the other mode, with $t' > t$ or $c \neq c'$. Whatever the mode considered, if node v is scheduled in slot t and on channel c, it means that no node $w \in Conflict(v)$ is scheduled in slot t and on channel c. Since both modes schedule nodes in $Conflict(v)$ in the same priority order and assign them the first available channel in the current time slot, they should reach the same decision: a contradiction. Since both modes apply the same rule to provide the next wave from the fist one, they provide the same assignment. □

8.3.5 Message complexity in the two modes of WAVE

We compare the centralized and distributed modes of WAVE in terms of number of messages needed to establish the collision-free schedule. We compute this number in the different phases (Neighborhood discovery, routing tree building and schedule establishment) of network operation.

8.3.5.1 Neighborhood discovery

Each node broadcast a $Hello$ message and receives $Hello$ messages from its neighbors. The centralized and distributed modes need the same number of $Hello$ messages.

8.3.5.2 Routing tree construction

Both centralized and distributed modes require the same number of messages to build the routing tree as it is an input for both of them.

8.3.5.3 Scheduling establishment

For each node u, we denote by $depth(u)$ its depth in the routing tree. Moreover, let us design by $Neigh(u)$ the list of one-hop neighbors of u. $V = |Neigh(u)|$. In addition, $MaxDepth$ denotes the depth of the routing tree. In the centralized mode as in the distributed mode, if the the size of a message is not compatible with the maximum size allowed by the standard MAC protocol, this message is fragmented.

- Centralized mode
  - Each node $u \neq$ sink, whose depth is $depth(u)$ transmits the list of its neighbors and its traffic demand $Gen(u)$ to the sink. This message needs $depth(u)$ hops to reach the sink. In total, we have:

$$\sum_{u \neq sink} depth(u)\ transmissions = averageDepth * (N-1)\ transmissions \quad (8.2)$$

  - The sink computes the scheduling and broadcasts it to sensor nodes. Notice that it is sufficient to send only the first wave and the repetitive factor of each slot. This message is broadcast to $MaxDepth$ hops.

  Thus, the total number of messages required to establish the scheduling in the centralized mode is:

$$averageDepth*(N-1)transmissions+the\ schedule\ message\ broadcast\ to\ MaxDepth\ hops \quad (8.3)$$

- Distributed mode
  - Computation of $Trans(u)$= the priority of node u:
    Each node $u \neq$ the sink transmits to its $Parent(u)$ the value of $Trans(u)$. This latter is equal to $Gen(u) + \sum_{v \in Child(u)} Trans(v)$. So we have $(N-1)$ transmitted messages.
  - Establishment of the first wave:
    1. Assignment of time slot and channel to nodes $v \in Conflict(u)$:
       If the immediate acknowledgement policy is adopted, each node $u \neq$ sink should notify its priority with nodes that are one hop away from u and nodes that are one hop away from $Parent(u)$. We assume that the priority of node u and its one-hop neighbors is included in the $Hello$ message. hence, no additional message is needed. After that, node u needs to notify its slot to its conflicting nodes. Therefore, we need $1 + V + (V-1) = 2V$ messages. Since we have $(N-1)$ nodes $\neq$ sink, we need a total of $2V * (N-1)$ messages in this phase.
    2. Computation of the total number of slots in the first wave:
       Each node $u \neq$ the sink transmits to its $Parent(u)$ the list of $MaxTrans(t)$ for each slot t known by u or its descendants. So this requires $(N-1)$ messages. Next, the sink computes the $MaxTrans(t)$ for each slot t and broadcasts a message that includes $T = total\ number\ of\ slots\ for\ the\ first\ wave$ and $MaxTrans(t)$, $\forall t : 1 \leq t \leq T$. This schedule message is broadcast to $MaxDepth$ hops.
    3. Computation of the total number of slots for each wave:
       Each node computes locally its slots for transmission and its slots for reception from its children. No message is needed.

Thus, the total number of messages required to establish the scheduling in the distributed mode is:

$$(2V+2)*(N-1)messages+one\ schedule\ message\ broadcast\ to\ MaxDepth\ hops \quad (8.4)$$

From Equations 8.4 and 8.3, the centralized mode of $WAVE$ outperforms the distributed one in terms of message complexity only if:

$$\boxed{averageDepth \leq 2V + 2}$$

8.3.5.4 Examples

• Example 1:
Let us consider a tree topology with 121 nodes, a maximum depth of 4 and a branching factor of 3.
We have $V = 1.99$, $averageDepth = 3.52$

$$3.52 \leq (2 * 1.99 + 1)$$

The centralized mode of $WAVE$ leads to the smallest number of messages.
• Example 2:
Let us consider a tree topology with 511 nodes, a maximum depth of 8 and a branching factor of 2.
We have $V = 1.99$, $averageDepth = 7$

$$7 > (2 * 1.99 + 1)$$

In this topology, the distributed mode of $WAVE$ is better.

8.4 DiSCA: an optimized version of WAVE

DiSCA inherits the major principles of WAVE. Nevertheless, DiSCA alleviates the stiffness of the pattern. Indeed, DiSCA tries to schedule the node having highest priority among its conflicting nodes, not in the current wave but in the earliest time slot where this node has at least one packet in its buffer. In the following, DiSCA operation will be detailed. Let us consider the following topology depicted in Figure 8.4. The sink is equipped with two radio interfaces. The scheduling obtained by $WAVE$ and DiSCA are illustrated respectively by Table 8.1 and Table 8.2.

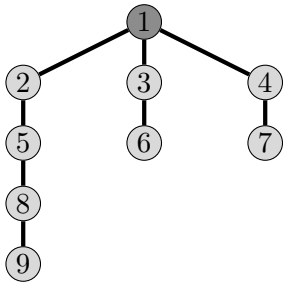


Figure 8.4: Topology illustrating the difference on schedule obtained by $WAVE$ and DiSCA.

Table 8.1: $WAVE$ scheduling with two channels

Channel \ Slot	1	2	3	4	5	6	7	8	9
1	2→1	5→2	4→1	2→1	5→2	4→1	2→ 1	5→2	2→1
	6→3	3→1			3→1				
	7→4								
2	8→5	9→8		8→5					

Table 8.2: DiSCA scheduling with two channels

Channel \ Slot	1	2	3	4	5	6	7	8
1	2→1	5→2	4→1	2→1	5→2	4→1	2→ 1	2→1
	6→3	3→1	5 →2		3→1			
	7→4							
2	8→5	9→8		8→5				

The first wave in $WAVE$ contains the first three slots (black cells) as illustrated in Table 8.1. $MaxTrans(1) = 4$, $MaxTrans(2) = 3$ and $MaxTrans(3) = 2$. Hence, the total number of slots needed to complete convergecast is $4 + 3 + 2 = 9$ slots. Nevertheless DiSCA needs only 8 slots. Indeed, node 5 has received a packet in slot 1 from node 8. DiSCA schedules the second transmission of node 5 not in the second wave but in the earliest slot available. Thus, node 5 is scheduled in the first wave. More precisely, it is scheduled in slot 3 as highlighted Table 8.2. The third transmission of node 5 is shifted also by DiSCA in an earlier slot (slot 5) as depicted in Table 8.2.

8.4.1 Principles

DiSCA obeys to the same rules as $WAVE$ rules, detailed in Section 8.3.2. Nevertheless, it adds the following rule:
Rule R'4: the i^{th} transmission of u **is scheduled in the first slot** t **where** u **has a packet to transmit** and on the first channel c (channels being visited in round robin) where the following conditions are met:

- u has an available interface,
- the parent of u has an available interface,

- the transmission of u does not conflict with the transmission of any node already scheduled in the slot t and on the channel c.

8.4.2 DiSCA distributed algorithm

Differences in pseudocodes between DiSCA and $WAVE$ are highlighted in red color.

Algorithm 10 processRecSlotChAssigned Procedure ($SlotChAssigned$)

1: **Input:** message $SlotChAssigned(slot, ch, v, Parent(v))$
2: **if** $v \in Conflict(u)$ **then**
3: update $ScheduledNodes$
4: **else if** $v = Child(u)$ **then**
5: update $AvailInterf(u, slot)$
6: update $EarliestPcktSlot(u)$
7: **else if** $(v = Parent(u))$ || $(Parent(v) = Parent(u)$ & $v \neq u)$ **then**
8: update $AvailInterf(Parent(u), slot)$
9: **end if**

Algorithm 11 ScheduleNode Function (v, $LastTxSlot(v)$,$EarliestPcktSlot(v)$, $Conflict(v)$, $nchannel$, $ScheduledNodes$)

Input: node v, $LastTxSlot(v)$: the last slot assigned to v, $EarliestPcktSlot(v)$: the smallest slot in which v has generated or received a packet not yet transmitted, $Conflict(v)$: the set of conflicting nodes, $nchannel$: the number of channels and $ScheduledNodes$: the table of scheduled nodes per slot and channel.
Output:
- $slot$: the slot assigned to node v
- c: channel allocated to node v
- $ScheduledNodes$: nodes scheduled per slot and channel.

Initialization:
$c \leftarrow 1$
$tx \leftarrow false$
$nChannelReached \leftarrow false$
$t \leftarrow max(LastTxSlot(v), EarliestPcktSlot(v))$

while $tx == false$ **do**
  while $(AvailInter(v,t) = 0 \;||\; AvailInter(Parent(v),t) = 0\;)$ **do**
    /*find a time slot with available interface for v & $Parent(v)$*/
    $t \leftarrow t + 1$
  end while
  repeat
    if $(Conflict(v) \bigcap ScheduledNodes(t,c) = \emptyset)$ **then**
      /*Node v can be scheduled */
      $tx \leftarrow true$;
      Node v transmits in slot t on channel c
      $AvailInter(v,t) \leftarrow AvailInter(v,t) - 1$
      $AvailInter(Parent(v),t) \leftarrow AvailInter(Parent(v),t) - 1$
      $ScheduledNodes(t,c) \leftarrow ScheduledNodes(t,c) \cup \{v\}$
    else
      if $(c < nchannel)$ **then**
        $c \leftarrow c + 1$ // try the next channel
      else
        $nChannelReached \leftarrow true$
      end if
    end if
  until $(tx \;||\; nChannelReached\;)$
  if $(tx == false)$ **then**
    $t \leftarrow t + 1$ // no channel available for v in slot t
  end if
end while
return (t, c, $ScheduledNodes$)

Algorithm 12 DiSCA algorithm: distributed mode

Input: $nchannel$: the number of channels, T: the routing tree where the local node u has a set of conflicting nodes $Conflict(u)$, $AvailInter(u)$: the number of available radio interfaces and a priority $Trans(u)$=number of packets that node u should transmit.
Output: $ScheduledNodes$: Channel and time slot assignment for local node u and $Conflict(u)$
Initialization:
$\forall v \in Conflict(u)$, $Slot(v) \leftarrow 0$ /*number of slots already assigned to node v*/
$LastTxSlot(u) \leftarrow 0$ /*the last slot assigned to u*/
$EarliestPcktSlot(u) \leftarrow 1$ /*the smallest slot in which u has generated / received a packet not yet transmitted.*/

Channel and slot assignment for node u
$SortedNodesList(u) \leftarrow$ the set $\{u\} \cup Conflict(u)$ sorted by decreasing priorities.
while $\exists v \in Conflict(u) such that Slot(v) < Trans(u)$ **do**
  if reception of a $SlotChAssigned(slot, c, v, Parent(v))$ Message **then**
    $processRecSlotChAssigned()$ /*process the received information
    $forwardRecSlotChAssigned()$ /*forward the received information
  end if
  if $(Slot(u) < Trans(u)$) & ($\forall v \in SortedNodesList(u)$ such that $Trans(v) > Trans(u)$, we have $Slot(v) > Slot(u))$ **then**
    ($slot$,c,$ScheduledNodes$)=ScheduleNode(u, $LastTxSlot(u)$, $EarliestPcktSlot(u)$, $Conflict(u)$, $nchannel$, $ScheduledNodes$)
    update of $LastTxSlot(u)$
    update of $EarliestPcktSlot(u)$
    $Slot(u) \leftarrow Slot(u) + 1$
    transmission of the $SlotChAssigned(slot, c, u, Parent(u))$ Message to the 1-hop neighbors of u.
  end if
end while

8.5 Comparative performance evaluation of WAVE and DiSCA

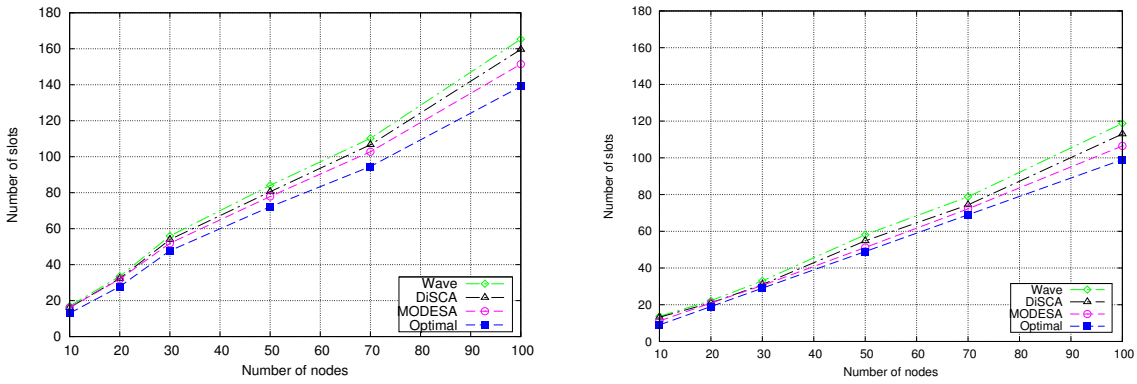
With our simulation tool based on GNU Octave [Octave], we evaluate the performance of $WAVE$ and DiSCA (i.e. number of slots required) in various configurations and compare them to the optimal one and our centralized solution MODESA. The number of nodes ranges from 10 to 100. Node 1 denotes the sink, equipped with 1, 2 or 3 radio interfaces. All other nodes have a single radio interface and generate traffic toward the sink. The number of available channels is 2 or 3. The random tree topology is generated according to the Galton-Watson branching stochastic process: any node has at most 3 children. Each result depicted is averaged on 20 simulations for topologies with less than 30 nodes and on 100 for others. We will distinguish two types of configurations: T_t configurations where the number of slots is dictated by the most demanding subtree and T_n configurations otherwise.

8.5.1 Homogeneous traffic

In this first series of simulation, each sensor node generates one packet per data gathering cycle

8.5.1.1 Comparison of $WAVE$ and DiSCA with the Optimal schedule and MODESA

Assuming that any node generates one packet and 2 channels are available at each node, we compare the number of slots provided by the optimal schedule, MODESA, $WAVE$ and DiSCA. We notice that T_t configurations are more greedy than T_n ones: T_t configurations of 100 nodes need in average 175 slots with $WAVE$ whereas T_n configurations need only 120 slots (see Figure 8.5). $WAVE$ is 18% (respectively 17%) away from the optimal in T_t configurations (respectively T_n configurations). Besides, $WAVE$ has slightly less performance than DiSCA but is much simpler. MODESA is 11% (respectively 10%) away from the optimal in T_t configurations (respectively T_n configurations) as depicted in Figure 8.5. DiSCA slightly outperforms WAVE.



(a) in T_t configuration

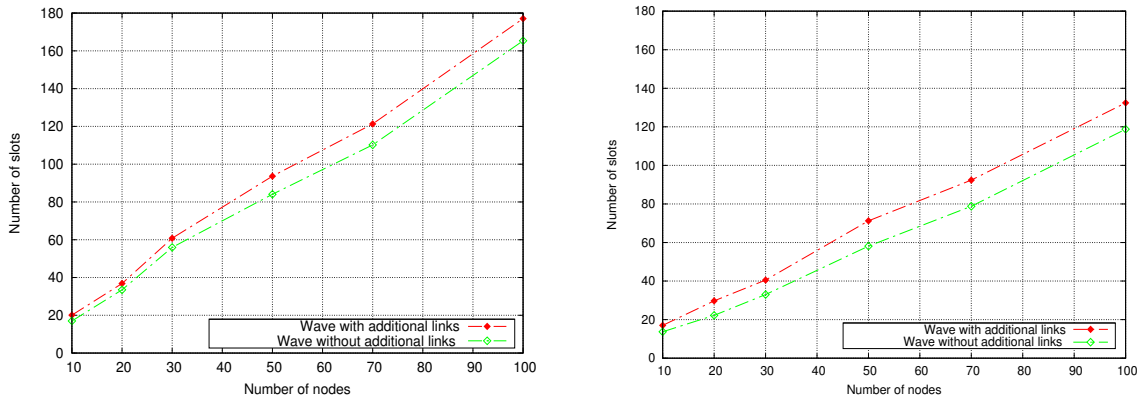
(b) in T_n configurations

Figure 8.5: $WAVE$ versus optimal schedule and MODESA: homogeneous traffic

8.5.1.2 Impact of additional links on $WAVE$

In previous simulations, we focused on computing spatial-reuse TDMA schedules once interfering links, that do not belong to the routing tree, have been eliminated. This is a crucial assumption for the work of Incel et al. [Durmaz Incel 2008]: their scheduling is applied after a channel allocation step that eliminates most of the interferences. Nevertheless, it was proven that [Ghosh 2009] assigning a minimum of channels to receivers such that all interfering links are removed, is NP-complete. It is important to note that our proposed algorithm $WAVE$ does not require that all interfering links are removed. It easy takes into account the presence of additional interfering links. Indeed, it is a black box for the algorithm.

With the same previous parameters, we add links to the links of the routing tree: for each node at even depth d in the tree, an additional link is generated with a node at depth $(d-1)$ different from its parent. Furthermore, with a probability equal to 0.5, another link is added with a node of depth $(d+1)$ different from its children. In average, 60% additional links are added.



(a) in T_t configuration

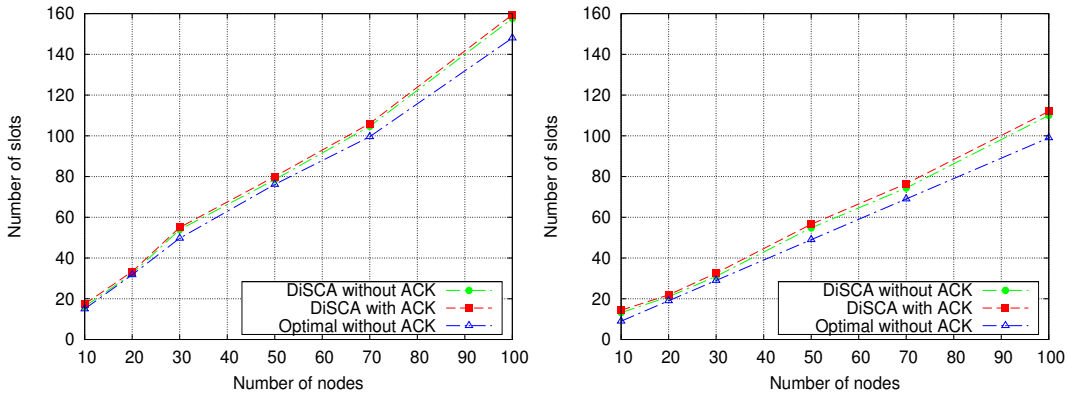
(b) in T_n configurations

Figure 8.6: $WAVE$ with additional interfering links

The existence of topology links that do not exist in the routing tree induces more conflicts. Hence the possibilities for parallel transmissions are reduced leading to a higher number of slots. As illustrated in Figure 8.6, for 100 nodes, added links result in 8% (respectively 11%) extra slots for $WAVE$ in T_t configurations (respectively T_n configurations).

8.5.1.3 Impact of immediate acknowledgment on DiSCA

We focus on DiSCA to investigate the impact of immediate acknowledgement. As expected the number of slots required when the immediate acknowledgement is used is higher than without (see Figure 8.7). However, the gap remains small thanks to the accurate definition of conflicting transmissions provided in Chapter 4 (less than 3%).



(a) in T_t configuration (b) in T_n configurations

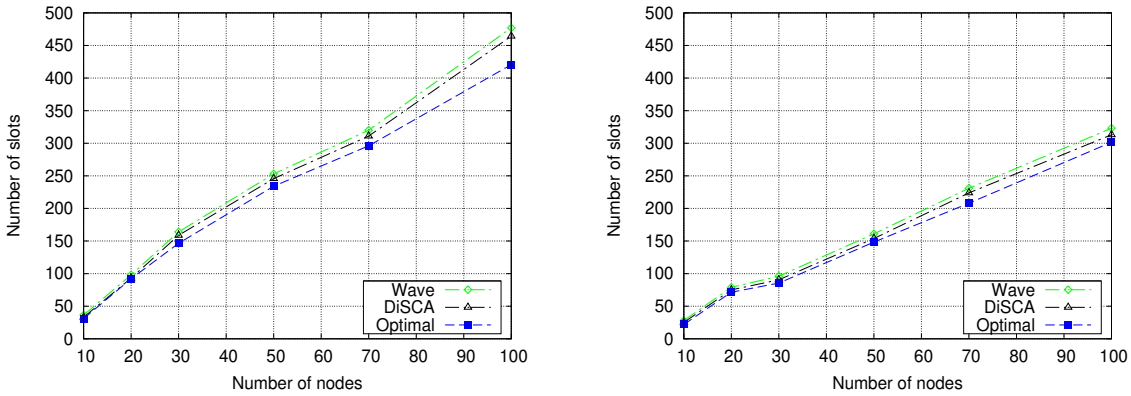
Figure 8.7: Impact of immediate acknowledgment on schedule length provided by DiSCA

8.5.2 Heterogeneous traffic

In this second series of simulations, we study the behavior of the proposed solution under heterogeneous nodes traffic. Each node generates a random number of packets between 1 and 5.

8.5.2.1 Comparison of $WAVE$ and DiSCA with the Optimal schedule and MODESA

The sink is equipped with one radio interface and three channels are available at each node. We notice in Figure 8.8 the same trend of curves as Figure 8.6 when nodes generate a single packet. $WAVE$ is 13% (respectively 11%) away from the optimal in T_t configurations (respectively T_n configurations). DiSCA still outperforms $WAVE$.

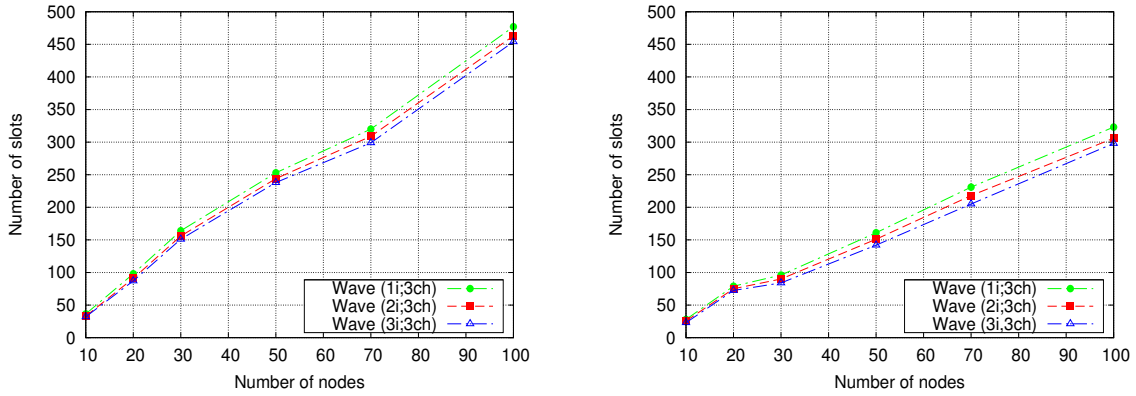


(a) in T_t configuration (b) in T_n configurations

Figure 8.8: $WAVE$ versus optimal schedule: heterogeneous traffic

8.5.2.2 Impact of multi-radio sink on $WAVE$

As illustrated in Figure 8.9, the number of slots to complete convergecast is reduced and so we have shorter convergecast delays. Indeed, when passing from $(1interface; 3channels)$ to $(3interfaces; 3channels)$, the number of slots is reduced by 6% (respectively 13%) in T_t configurations (respectively T_n configurations). This can be explained by the fact that in T_t configurations, the dominating subtree is the only subtree scheduled in the last time slots so the number of radio interfaces has no effect. Nevertheless, in T_n configurations, the demand is balanced among subtrees and the sink can receive simultaneously from its children even in the last time slots.



(a) in T_t configuration

(b) in T_n configurations

Figure 8.9: Impact of multi-radio sink on number of slots.

8.6 Conclusion

This chapter tackled, scalability, the last issue addressed in Part III. We presented a distributed TDMA-based scheme called $WAVE$ that jointly considers channel allocation and node scheduling for data delivery from sensor nodes to the sink. $WAVE$ is simple to implement and efficient. It is based on successive waves paradigm. Indeed, in each wave, each node having a packet to transmit is assigned a time slot and a channel. The first wave constitutes the pattern. Each next wave is an optimized subset of the first wave: only the slots that will contain transmissions are repeated and they always occur in the same order as in the first wave. Hence, all slots allocated are not empty. In addition, we proposed DiSCA, an optimized version of WAVE that provides slightly shorter schedules than WAVE. Nevertheless, DiSCA is more complex than WAVE in terms of number of exchanged messages and computations.

Moreover, we compared the performance of our scheme with the optimal one and MODESA our centralized solution proposed in Chapter 5 in terms of number of slots needed to complete the convergecast. Simulations results revealed that our scheme is not far from the optimal bound for raw convergecast. Unlike most previously published works, $WAVE$ does not suppose that all interfering links have been removed by channel allocation. In addition,

$WAVE$ is able to easily adapt to traffic changes. $WAVE$ could be used to provide the schedule applied in the 802.15.4e TSCH [TG4e 2012].

Chapter 9

Conclusion and Perspectives

Contents

9.1 Synthesis 159
9.2 Perspectives 161

9.1 Synthesis

Wireless Sensor Networks (WSNs) arise from the human need of observing various environments. WSNs combine wireless communications with sensing technologies to transmit, using multi-hop paths, sensed data to the control center. Recently, WSNs are experiencing a tremendous industrial interest due to their advantages over wired networks in some scenarios. Industrial applications have peculiarities that distinguish them from classical ones. Although some requirements such as scalability and energy efficiency are shared with classical applications, stringent latency requirement and throughput are by far more critical. Moreover, WSNs in industrial environment have to be robust against interferences because they usually co-exist with other wireless technologies such as WiFi, Bluetooth, etc. Distributing the communication across multiple channels to mitigate interferences and to increase parallel transmissions is an attractive solution.

This thesis is particularly concerned with applications gathering non critical parameters (temperature, pressure, vibration, etc). In these latter, sensor nodes sense the surrounding environment and send their data, directly or via multiple hops, to on-board computers that process them. This many-to-one communication pattern is called convergecast and it is used in most industrial applications. A data convergecast protocol should ensure the freshness of these data reports when they reach the sink. That is why this kind of traffic is very delay-sensitive. To minimize end-to-end delay, sensors should deliver their data as fast as possible to the sink. The intrinsic characteristics of WSNs such as scarce energy budget coupled with channel contention and interferences, raise great challenges with regard to energy efficiency.

Moreover, the new MAC amendment, *IEEE* 802.15.4*e* TSCH [TG4e 2012], redesigns the existing *IEEE* 802.15.4 and adds functionality to better meet industrial markets requirements. The TimeSlotted Channel Hopping (TSCH) mode ensures robustness and high reliability against interferences by channel hopping. *IEEE* 802.15.4*e* highlights how the MAC layer executes the nodes schedule. Nevertheless, the policy that provides such a schedule is not specified.
That is why, this thesis provides scheduling solutions for data converegacast applications that require timely and efficient delivery of collected data in multichannel WSNs.

To understand properly the specific energy constraints and the major issues and advantages of multichannel communications in WSNs, we provided a comprehensive study of energy efficient techniques and channel assignment protocols in Part I.

In the next step, we have examined, from a theoretical point of view, the problem of raw data convergecast in multichannel WSNs. Specifically, we provided a very accurate definition of conflicting transmissions for data gathering. Furthermore, we derived the minimum delay for collecting sensor data in various topologies such as line, multiline and trees and gave corresponding optimal scheduling algorithms. Our analysis is applied to sink equipped with multiple radio transceivers. This theoretical study helped us to evaluate how close the solutions we propose are to the established theoretical lower bounds.

We tackled in Part II, static environments where sensors nodes are static within structural surfaces for example and the wireless environment is known. We proposed centralized solutions for joint time slot and channel assignment that achieve performances near to optimal schedules. Indeed, MODESA, our optimized multichannel slot assignment for raw convergecast, orchestrates nodes activities in a contention-free manner. MODESA is traffic aware and supports different channel allocation strategies. Conducted simulations have corroborated the efficiency of our technique in comparison to TMCP, a well-known cluster-based convergecast protocol, and shown that MODESA is closer to the optimal schedule and uses low buffer size. That is why MODESA is suitable for applications that require short data delivery delays. Moreover, unlike the work of Incel [Incel 2012], MODESA does not require that the convergecast tree links are the only links present in the topology. Merging multichannel communications and multipath routing was also investigated. Simulations results depict the contribution of multipath routing to delays minimization. In addition, available paths can be used to balance network load among nodes specifically in data intensive applications.

On the other side, some data gathering applications collect data toward multiple sinks to ensure more reliable data gathering (path diversity) and or traffic differentiation to the flows. We extended MODESA to the context of multiple sinks gathering applications. Hence, we proposed MUSIKA to derive collision-free schedules for raw data convergecast with the smallest schedule length. This contribution presents the first attempt to investigate the concept of overlapping interference-free convergecast trees in multichannel WSNs.

In addition, retransmissions or sudden changes in traffic loads necessitate an updated allocation of slots and channels changes. Thus, generating a new schedule for the entire network especially in large scale WSN, is a taunting task. That is why we proposed an adaptive slot allocation solution based on an incremental technique and called AMSA. It

unifies the management of additional slots, whatever their origin: changes in the application demands or in the medium access demands due to retransmissions. Simulations shown that AMSA ensures efficient convergecast especially for low (5%) and moderate (10%) additional transmissions. Furthermore AMSA schedule length is less than 10% higher than the optimal one. In case of high (> 20%) additional transmissions, recomputing the schedule for all nodes is more efficient. However, it requires a higher number of messages to reschedule the whole network specifically in large scale WSNs.

Therefore, we proposed, $WAVE$, a distributed wave-based scheduling for raw convergecast applications in multichannel WSNs. $WAVE$ builds the first wave granting to a node the smallest available time slot and channel. This first wave is repeated, after optimization, the number of times requested by the most demanding node. Simulations results confirmed the efficiency of $WAVE$ and demonstrated the trade-off between simplicity and fast convergecast.

While preparing this dissertation, we learned several lessons which we will summarize in five points:

✓ **Lesson 1:** Multichannel communications in WSNs have their specificities that distinguish them from other networks. Hardware, overhead and processing constraints of sensors should be taken into account. *Semi-dynamic channel assignment allows the adaptivity to traffic changes and interferences while copying with long switching delays.*

✓ **Lesson 2**: The minimum time to complete the delivery of packets in a convergecast application where data cannot be aggregated, depends on the routing topology. *Balanced routing trees in terms of nodes traffic demands, achieve delivery delays shorter than unbalanced trees.*

✓ **Lesson 3:** *Multipath routing is an elegant way to provide shorter schedules length and load balancing* by splitting traffic over several paths.

✓ **Lesson 4:** *To achieve energy efficiency, we need to combine several energy efficient techniques.* In our work, node activity scheduling, multichannel communications and multipath routing were combined to provide fast data gathering while preserving the scarce energy budget of sensors.

✓ **Lesson 5:** *Channel allocation strategy depends on the context:* a greedy strategy favors the channel with the best quality whereas a round robin strategy provides a better load balancing.

9.2 Perspectives

The algorithms, devoted to fast raw convergecast in multichannel WSNs presented in this manuscript, have shown a good performance. Nevertheless, several future research directions open up.

- *More realistic models:*
  Our simulation results have revealed the contribution of the multichannel paradigm on the improvement of network capacity and minimization of data convergecast delays.

Nevertheless, there is a gap between simulations and reality. We envision to study MODESA and $WAVE$ implemented on a real WSN operating in conditions representative of their real environment.

- *Incorporation with $IEEE$ 802.15.4e TSCH:*
  The effectiveness of MODESA and $Wave$ has been demonstrated using simulations results. Nevertheless, to incorporate MODESA and $Wave$ with the $IEEE$ 802.15.4e TSCH, a study should be done to specify if the information needed to establish the schedule, can be supported by the new MAC amendment.

- *Real-time applications:*
  Multimedia applications, such as video or voice-based applications, require real-time data delivery. Multichannel communications help to reduce the delay by increasing the number of parallel transmissions. However, some scheduled messages have a strict deadline in order to take appropriate actions in real time. Priorities could be assigned to packets. They indicate for each device if the arriving packet has to be transmitted directly or it can be buffered.

- *Coexistence of aggregated convergecast and raw data convergecast:*
  The three-level architecture proposed in Chapter 3 can combine the two modes of data convergecast: raw and aggregated. In each cluster, sensor nodes send their data to the aggregator. Then, each aggregator transmits, in a raw data convergecast fashion, aggregated data to the sink. It would be interesting to study how the solution proposed in this thesis could be extended to address the coexistence of these two modes of convergecast.

- *Adaptivity to environmental and application changes:*
  While multichannel WSNs are envisioned for many applications, the unreliable nature of wireless network, the node mobility and the changes of traffic profiles, make their design non trivial. In such applications, the channel assignment needs to adapt to changing channels and applications conditions. This leads to the urgent necessity of mechanisms that quickly understand and reconfigure the WSN to ensure the maximum possible throughput. Techniques like adaptive learning that learn from the history and act to maximize the reward, can be envisioned [Phung 2012, Phung 2013]. Moreover, game theory [Chen 2011] is also a promising approach provided that the convergence is quickly obtained.

- *Cognitive radio sensor networks:*
  WSNs operate on the unlicensed spectrum that includes essentially the ISM band. This latter is shared by many technologies such as 802.11, 802.15.1 and 802.15.4. The coexistence of WSNs with WiFi and Bluetooth can drastically degrade the network performances. On the other side, the licensed spectrum is for the exclusive use of designated users. Unfortunately, the unlicensed spectrum bands are becoming scarce, while large portions of the entire radio spectrum remain unused. We think that cognitive

radio technology [Akan 2009] allows WSN to provide access to a new spectrum with better characteristics. Hence, channel assignment approaches, should be adapted to take into account these new available bands. It would be interesting to study how the solution proposed in this thesis could be extended to address this problem. A first reflection has been published in [Mabrouk 2014].

Appendix A

List of Publications

1. **Journal Papers:**

   - **R. Soua**, E. Livolant, P. Minet, "Adaptive Strategy for an Optimized Collision-Free Slot Assignment in Multichannel Wireless Sensor Networks", the Journal of Sensors and Actuators Networks (JSAN). Accepted June 2013.
   - **R. Soua**, P. Minet, "Multichannel Assignment Protocols in Wireless Sensor Networks: A Comprehensive Survey", submitted to the Pervasive and Mobile Computing Journal (PMC)

2. **Conference Papers:**

   - **R. Soua**, P. Minet, E. Livolant, "DiSCA: a Distributed Scheduling for Convergecast in Multichannel Wireless Sensor Networks", submitted to The 28th IEEE International Conference on Advanced Information Networking and Applications (DCOSS 2014).
   - **R. Soua**, P. Minet, E. Livolant, "A Distributed Joint Channel and Time Slot Assignment for Convergecast in Wireless Sensor Networks", accepted, The Sixth IFIP International Conference on New Technologies, Mobility and Security (NTMS 2014).
   - O. Mabrouk, H. Idoudi, I. Amdouni, **R. Soua**, P. Minet, L. Saidane, "OTICOR: Opportunistic Time Slot Assignment in Cognitive Radio Sensor Networks", accepted, The 28th IEEE International Conference on Advanced Information Networking and Applications(AINA-2014), Canada, May 2014.
   - **R. Soua**, E. Livolant, P. Minet, "MUSIKA: a Multichannel Multi-sink Data Gathering Algorithm in Wireless Sensor Networks", the 9^{th} IEEE International Wireless Communications & Mobile Computing Conference (IWCMC), July, 2013.
   - **R. Soua**, P. Minet, E. Livolant, "MODESA: an Optimized Multichannel Slot Assignment for Raw Data Convergecast in Wireless Sensor Networks", the the 31^{st} International Performance Computing and Communications Conference (IPCCC), Austin, Texas, December 2012.

- I. Amdouni, **R. Soua**, E. Livolant, P. Minet, "Delay Optimized Time Slot Assignment for Data Gathering Applications in Wireless Sensor Networks", The third International Conference on Wireless Communications in Unusual and Confined Areas, Clermont-ferrand, France, August 28th-30th, 2012.
- **R. Soua**, P. Minet, "A Survey on Multichannel Assignment Protocols in Wireless Sensor Networks", IFIP Wireless Days, Niagara Falls, Ontario, Canada, October 10-12, 2011.
- **R. Soua**, P. Minet, "A Survey on Energy Efficient Techniques in Wireless Sensor Networks", IWMNC 2011, Toulouse, France, October 26-28,2011.

Appendix B

Résumé

B.1 Contexte et motivations

Les réseaux de capteurs sans fil (RCsF) permettent une observation fine de leur envionnement. La portée de transmission réduite des capteurs ainsi que la bande passante limitée ont pour conséquence des communications multi-sauts et à faible débit. Les noeuds surveillent leur evironnement et transmettent leurs données, directement ou via plusieurs sauts, à une entité centrale appelée puits. Dans certaines applications de collecte de données, les noeuds intermédiaires (les noeuds situés entre le noeud source et le puits) ne font pas d'agrégation. Cette collecte de donnée est appelée collecte de données sans agrégation. Dans ce contexte, les noeuds près du puits transmettent plus de messages que les noeuds qui sont distants du puits. Par conséquence, les noeuds proches du puits doivent avoir des opportunités d'accès au canal proportionnelles à la quantité des données générées et relayées. La collecte de données est une opération clé dans les environnements industriels. Ainsi par exemple, dans les centrales électriques et les avions, des centaines de capteurs sont ou seront déployés dans des milieux confinés afin de remonter des mesures de pression, température, vibration, etc. Remarquons cependant qu'il ne s'agit pas de données critiques, mais de données utiles à la maintenance par exemple.

Deux problèmes sont suscités par la collecte de données: (1) la garantie de remise et minimisation des délais de collecte des données (2) l'efficacité énergétique. De courts délais garantissent la fraîcheur des données délivrées au puits. En plus, la garantie de remise permet une surveillance plus précise de l'environnement contrôlé.

Un problème majeur rencontré par les algorithme de collecte de données dans les RCsF, est les interférences. Pour contourner ce problème, les chercheurs ont recours aux communications multicanal. En effet, l'aspect multicanal est utilisé pour augmenter le nombre de transmissions en parallèle, la robustesse face aux perturbations internes et externes. Par conséquente, le multicanal permet d'améliorer les performances de collecte de données en termes de débit supporté, délai et taux de remise. Vu que la collecte de données implique un grand nombre de noeuds qui transmettent simultanément, les collisions et donc les retransmissions sont fréquentes et représentent un obstacle pour avoir de délais de remise bornés. En effet, les collisions engendrent la perte des données. Les retransmissions augmentent les délais de

remise des paquets. Contrairement aux protocoles d'accès au médium à contention qui souffrent de l'inefficacité à cause des backoff et des collisions, les méthodes d'accès au médium sans contention garantissent des délais bornés. Ces protocoles encore appelés protocoles d'accès au médium déterministes, assurent que toute transmission d'un noeud n'interfère pas avec toute autre transmission simultanée. Ceci est obtenu par l'allocation de canaux et de slot temporels aux noeuds de telle manière que les interférences sont évitées. Ainsi, il est facile de contrôler les délais des paquets pour atteindre le puits. En outre, les protocoles sans contention sont plus économe en énergie que les protocoles basés sur la contention: ils éliminent les principales sources de gaspillage d'énergie que sont l'écoute passive, overhearing et les collisions.
De plus, un noeud est actif seulement s'il transmet à son parent ou il reçoit de ses enfants. Les noeuds éteignent leur radio sinon. Par conséquent, les protocoles sans collision sont idéales pour les réseaux de capteurs où les noeuds sont alimentés par batterie.

Les accès au médium sont organisés selon une supertrame. Cette supertrame est délimitée par une balise comprenant une période avec contention avec accès en CSMA/CA et une période sans contention où les accès sont effectués dans des slots selon un ordonnancement préétabli. Lorsque cette supertrame se répète, la durée entre deux balises successives détermine la longueur du cycle. La Figure 1 illustre la structure de la supertrame.

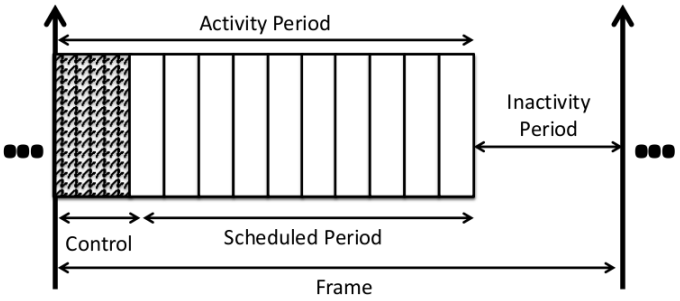


Figure B.1: Schéma d'une supertrame.

Notre objectif est de minimiser le nombre de slots nécessaires à la collecte de données. En effet, minimiser le nombre de slots permet en effet de diminuer les délais de collecte de données. Considérons une supertrame telle que chaque noeud dispose des slots nécessaires pour transmettre ses données locales et les données reçues de ses enfants. Le pire délai de remise est égal à la durée du cycle plus la durée de la période sans contention. Remarquons que par ailleurs, plus ce délai est petit, plus les données collectées dans cette supertrame seront cohérentes temporellement.

Le but de ce travail est de fournir des ordonnancements sans conflit qui orchestrent les activités des noeuds dans un environnent sans fil partagé tout en minimisant le délai nécessaire à la collecte de données. Ces solutions doivent être bien adaptées aux caractéristiques intrinsèques des réseaux de capteurs telles que l'efficacité énergétique, les contraintes temporelles, l'adaptabilité à l'environnement et le passage à l'échelle.

B.2 Allocation conjointe de slot et canaux dans les RCsF

B.2.1 Bornes théoriques

Nous abordons le problème du temps optimal nécessaire à la collecte de données sans agrégation dans un RCsF multicanal (un nombre de canaux $nchannel > 1$ est disponible sur chaque capteur). Nous adoptons un protocole d'accès au médium sans contention. Nous proposons une définition très précise des transmissions conflictuelles pour la collecte de données. Nous établissons aussi les bornes inférieures pour la collecte de données pour un puits équipé de multiples interfaces radio ($ninterf$ est le nombre d'interfaces radio du puits, $ninterf \geq 1$, pour les autres noeuds, $ninterf = 1$) et pour un trafic hétérogène et homogène ($Gen(u)$ est le nombre de paquets générés par u pour transmettre ses propres données).

B.2.1.1 Modélisation des interférences pour la collecte de données

Pour chaque noeud u, la détermination de l'ensemble des noeuds conflictuels avec u, noté $Conflict(u)$, dépend de la politique utilisé pour l'acquittement (sans ou avec acquittement immédiat).

Lemma 9. *Dans une collecte de données et en l'absence de l'acquittement, $Conflict(u)$ comprend: a) Le noeud u lui même, b) le noeud $Parent(u)$, c) tous les enfants de u, d) tous les noeuds qui sont à un saut de $Parent(u)$, e) tous les noeuds dont le parent est à un saut de u*

Lemma 10. *Dans une collecte de données avec acquittement immédiat, $Conflict(u)$ contient a) le noeud u lui même, b) le $Parent(u)$, c') tous les noeuds qui sont à un saut de u ou $Parent(u)$, d') tous les noeuds dont le parent est à un saut de u ou $Parent(u)$*

B.2.1.2 Bornes théoriques du nombre optimal de slots

Theorem 12. *Dans tout RCSF en arbre avec demandes hétérogènes des noeuds où chaque noeud dispose de nchannel > 1 canaux, le nombre minimum de slots nécessaires à une collecte de données sans agrégation est Max(S_n,S_t), avec $S_n = \lceil \frac{\sum_{u \in WSN} Gen(u)}{g} \rceil$, $g = min(nchild, nchannel, ninterf)$ et $S_t = Gen(ch1) + 2\sum_{v \in\ Subtree(ch1), v \neq ch1}(Gen(v)) + \delta$, où $\delta = 1$ si le $(g+1)^{th}$ enfant du puits a besoin du même nombre de transmissions que le premier, $\delta = 0$ sinon.*

B.2.1.3 Configurations T_t et T_n

Toute configuration est définie par une topologie (ensemble de noeuds et liens) et les demandes de slots de chaque noeud correspondant aux données générées localement. Nous définissons deux types de configurations, T_t et T_n:

- Une configuration est dite T_t si et seulement si le nombre optimal de slots est imposé par le sous-arbre le plus demandeur de slots, sous-arbre de racine un enfant i du puits. La demande du sous-arbre est égale à $Gen(i) + 2\sum_{v \neq i, v \in subtree(i)} Gen(v)$. Le nombre de slot dans une configuration T_t est donné par S_t.

- Une configuration est dite T_n si et seulement si le nombre optimal de slots dépend uniquement du nombre total de demandes des noeuds et de $g = min(nchild, nchannel, ninterf)$. Le nombre de slot est donné par S_n.

B.3 MODESA: algorithme optimisé pour l'allocation conjointe de slots temporels et canaux

Nous présentons ici notre algorithme centralisé d'allocation conjointe de slot et canaux. MODESA prend en compte la disponibilité de plusieurs canaux pour réduire la période d'activité dans la supertrame de collecte de données, tout en assurant un accès au médium proportionnel au trafic du noeud. MODESA construit l'ordonnancement dès qu'il dispose des informations nécessaires (e.g. liens de topologie, demandes des noeuds, arbre de routage, noeuds conflictuels).

MODESA construit l'ordonnancement multicanal slot par slot. A chaque itération, MODESA alloue le slot courant à un ou plusieurs tuples (numero de slot, émetteur, récepteur, canal) en appliquant les règles suivantes:
- Chaque noeud a une priorité dynamique. Cette priorité est égale à $remPckt(u) * Rcv(Parent(u))$ où $remPckt(u)$ est le nombre de paquets que le noeud a dans son buffer à l'itération courante. $Rcv(Parent(u))$ est le nombre total de paquets que le parent du noeud doit recevoir dans un cycle. Cette heuristique vise à réduire le nombre de paquets tamponnés en favorisant les noeuds qui ont des paquets à transmettre au parent devant recevoir un grand nombre de paquets.
- Seuls les noeuds ayant des données à transmettre entrent en compétition pour le slot courant.
- De plus, pour être autorisé à transmettre dans un slot, un noeud et son parent doivent avoir une interface disponible.
- Pour tout slot, le premier noeud ordonnancé est le noeud de plus forte priorité parmi tous les noeuds ayant des données à transmettre. Si plusieurs noeuds ont la même priorité, MODESA choisit le noeud de plus petit identificateur. Le noeud choisi est ordonnancé sur le premier canal disponible c.
- Tout noeud en compétition peut être ordonnancé dans le slot courant sur le canal c si et seulement s'il n'est pas conflictuel avec les noeuds déjà ordonnancés sur ce canal et dans ce slot.
- Un noeud conflictuel avec les noeuds déjà ordonnancés sur ce canal est ordonnancé sur un autre canal, s'il en existe. Sinon dans le slot suivant.

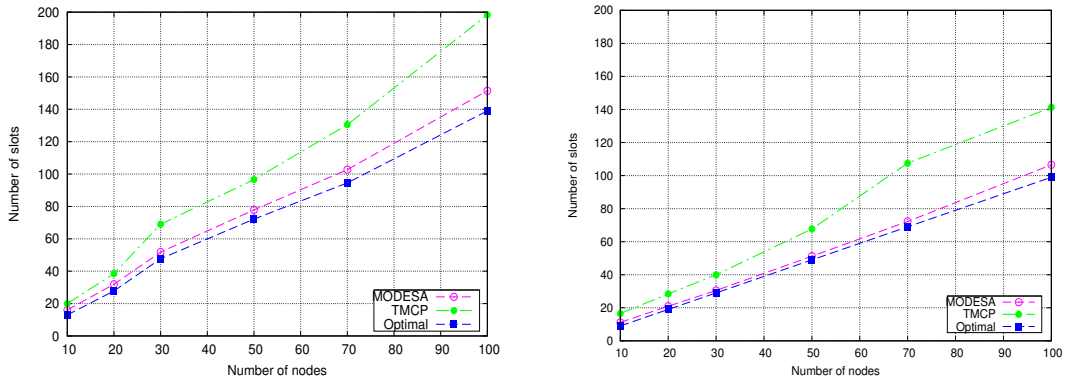
B.3.1 Optimalité de MODESA

Nous considérons des trafics homogènes où chaque noeud génère un message: $Gen(u) = 1$. Quelque soir $u \neq du\ puits$.

Theorem 13. *En environnement multicanal, nchannel > 1, MODESA est optimal pour des demandes homogènes dans des topologies linéaires, multi-lignes et les arbres équilibrés.*

B.3.2 Evaluation de performances

Nous avons comparé MODESA à TMCP, un algorithme bien connu d'allocation de slot temporel et de canaux pour la collecte de données. Les résultats de simulations ont montré que MODESA fournit de plus petits délais de collecte et a de plus faible exigences en tampons comme illustré respectivement dans les figures B.2 et B.3.



(a) (b)

Figure B.2: Ordonnacement avec plusiers canaux: performances de MODESA et TMCP en nombre de slots dans (a) configurations T_t (b) configurations T_n.

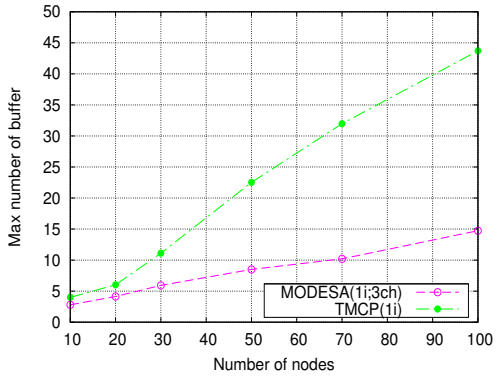
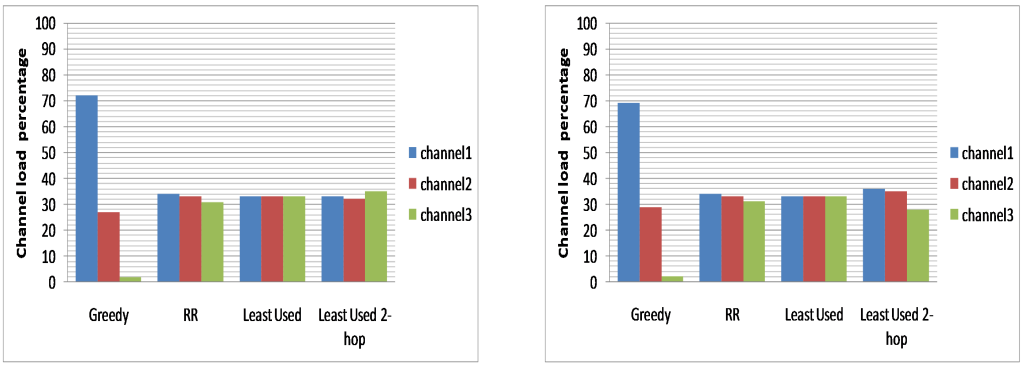


Figure B.3: Ordonnacement avec plusieurs canaux: performances de MODESA et TMCP en termes de tampons.

Nous avons aussi évalué des différentes stratégies d'allocation de canaux. La stratégie Round Robin offre le meilleur compromis entre la simplicité et l'équilibre de charge sur les canaux. Cette comparaison est illustrée dans la figure B.4.

La stratégie round robin offre le meilleur compromis entre simplicité et équilibrage de charges.



(a) (b)

Figure B.4: Charges sur les canaux dans des topologies de 100 noeuds et avec un puits équippé de a) 1 interface and 3 canaux (b) 2 interfaces and 3 canaux

B.4 MUSIKA: algorithme d'allocation conjointe de slots et de canaux pour les RCsF multicanaux multi-puits

Dans MODESA, nous nous sommes focalisés sur la collecte de données vers un seul puits. Toutefois, un RCsF peut comprendre plusieurs puits, pour réaliser des économies d'énergie, ou pour fiabiliser des données critiques (diversité de chemins) ou encore pour supporter plusieurs applications. Nous nous concentrons sur un contexte multicanal multi-puits avec un trafic dédié par puits.

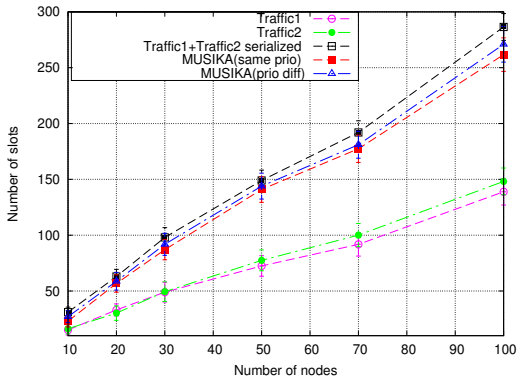
MUSIKA adopte le concept d'ordonnancement sans interférence des transmissions des arbres de collecte de données se recouvrant. Par conséquent, un noeud peut être impliqué dans plusieurs assignations de slot qui appartiennent à différents arbres de collecte. MUSIKA construit l'ordonnancement multicanal slot par slot en appliquant les règles suivantes:
- Seuls les noeuds qui ont au moins un paquet dans leurs tampons entrent en compétition pour le slot courant. Ils sont ordonnés selon leurs priorités. Soit N cet ensemble ordonné.
- Le noeud de plus haute priorité est sélectionné en premier.
- Un noeud autorisé à émettre dans le slot courant va transmettre le premier paquet dans sa file d'attente qui correspond à la classe de trafic la plus prioritaire. Dans le cas où plusieurs classe de trafic ont le même niveau d'importance, le premier paquet présent dans la plus longue file d'attente va être choisi.
- Un noeud est autorisé à émettre dans le slot courant si et seulement si:

  - Ce noeud et son parent dans l'arbre de collecte qui correspond à la classe de trafic du paquet choisi, ont chacun une interface radio disponible.
  - Il existe un canal où ce noeud n'est pas en conflit avec les autres noeuds déjà ordonnancés dans le même slot.

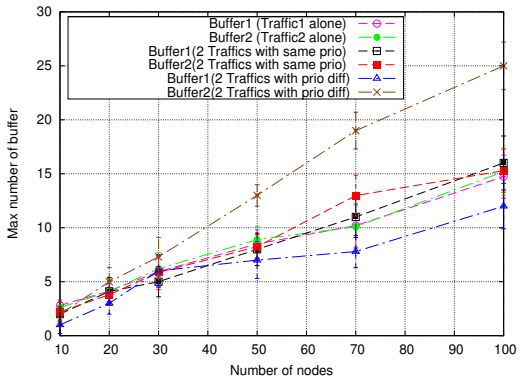
- Le prochain noeud sélectionné est le noeud suivant dans N. Il est autorisé à transmettre selon les règles 3 et 4 et ainsi de suite jusqu'à que tous les noeuds dans N n'aient plus aucun message à transmettre.

La priorité d'un noeud est calculée en tenant compte : (1) du nombre de paquets présents dans ses files d'attente pour éviter une saturation des tampons, (2) de la somme des paquets que son parent va recevoir dans un cycle de collecte de données, somme calculée sur tous les arbres de collecte (3) des classes des paquets à émettre par le noeud dans le slot courant pour fournir une différentiation de trafic si celle-ci est requise par l'application.

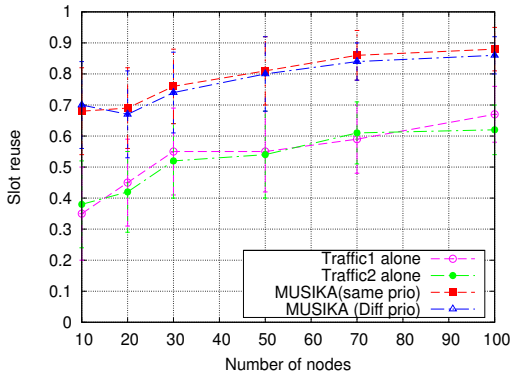
• Evaluation de performances:
D'abord, nous avons évalué le nombre de slots nécessaires pour la collecte de données. Nous avons distingué 2 cas: (1) tous les trafics ont la même priorité (2) un trafic a une plus grande priorité que les autres. Les résultats illustrés sur la figure B.5, montrent que le nombre de slot dont MUSIKA a besoin dans les deux cas est inférieur au nombre de slots lorsque les trafics sont ordonnancés en série.



(a) Nombre de slots



(b) Taille maximale du tampon



(c) Taux de l'utilisation du slot

Figure B.5: Performances de MUSIKA.

La figure B.5(b) montre que, lorsque les trafic ont la même priorité, MUSIKA a besoin d'un nombre de tampons très proche de celui nécessaire lorsqu'il y a un seul trafic. Cela s'explique par le fait que MUSIKA favorise les noeuds qui ont les plus longues files d'attente.

B.5 Adaptabilité et passage à l'échelle pour l'allocation conjointe des slots temporels et canaux

Bien que nous ayons bien étudié la collecte de données sans agrégation à la fois d'un point de vue théorique et d'un point de vue pratique dans les sections précédentes, la question sur le comportement des noeuds en présence de retransmissions ou de changements temporaires dans les besoins de l'application, reste sans réponse. En outre, tous les algorithmes d'allocation conjointe de slot et canaux que nous avons proposés sont centralisés. En effet, le puits doit calculer les ordonnancements et les diffuser aux capteurs. Dès que tous les capteurs ont reçu l'ordonnancement, ils l'appliquent. Dès que des changements ont lieu (ex. défaillance d'un noeud, nouveau trafic), le puits doit être informé de ces changements, recalculer un ordonnancement et le diffuser. Cette situation rend les algorithmes centralisés inefficaces dans les réseaux de capteurs à grande échelle. Ainsi, l'objectif de cette partie sera la proposition d'ordonnancement sans conflit adaptatifs et distribués.

B.5.1 Stratégie adaptative pour l'allocation conjointe de slot temporel et canaux

B.5.1.1 Problème

Dans de nombreux déploiements réels, le RCsF fait face à des requêtes dynamiques de transmissions (des alarmes, un trafic supplémentaire temporaire). Ainsi, l'algorithme d'attribution conjointe de slots temporels et de canaux doit s'adapter :

- à la retransmission des messages qui n'ont pas été acquittés par la couche MAC.
- aux changement temporaire du besoin de l'application en raison des alarmes, par exemple. Les alarmes sont associées à des exigences de délais forts et doivent être livrés de façon fiable au puits. Plusieurs possibilités existent:
  - si un slot a été affecté à un noeud émetteur, l'alarme est envoyé en premier, prenant la place des données régulières. Ces données régulières seront ensuite envoyés dans un slot supplémentaire, accordé par la solution adaptative.
  - si aucun slot a été accordé au noeud émetteur, dans le pire des cas, l'alarme est envoyée dans le cycle suivant qui suit le message de contrôle qui demande le transfert d'alarme.

Nous proposons une solution d'assignation conjointe de slots et de canaux basée sur une technique incrémentale. Ainsi après la construction d'un ordonnancement initial optimisé, ordonnancement dit primaire, seuls les données différentielles correspondant à une attribution temporaire de slot, sont diffusées par le puits aux capteurs.

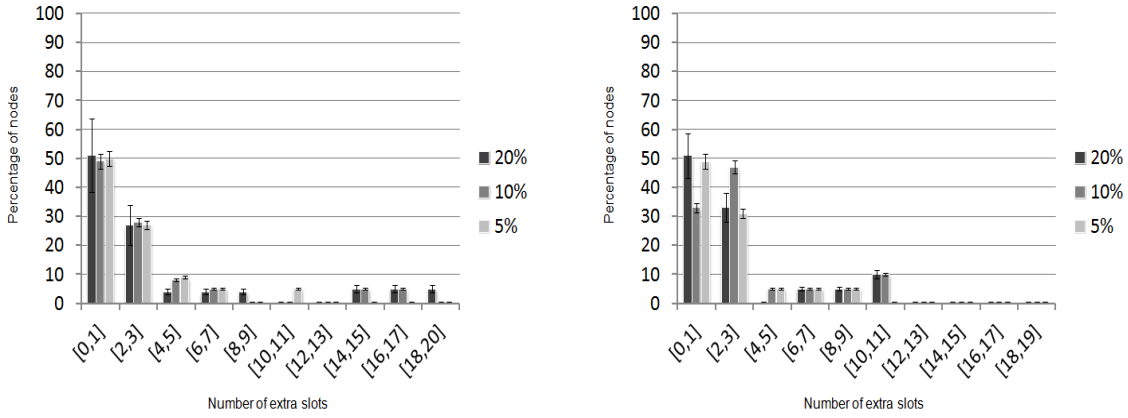
B.5.1.2 AMSA: solution incrémentale pour l'attribution adaptative conjointe de slot temporels et canaux

AMSA suit les règles ci-dessous:
- Chaque noeud qui détecte une modification de ses besoins de transmission envoie sa

demande de slots bonus au puits par un message de contrôle.
- Le puits essaie d'insérer les slots bonus pour répondre aux besoins supplémentaires dans l'ordonnancement primaire. Si c'est impossible, de nouveaux slots sont attribués dans la période d'inactivité. Le puits diffuse les mises à jour de cet ordonnancement dans l'arbre.
- Quand un noeud reçoit l'affectation du slot supplémentaire, il utilise (indifféremment) les slots réguliers (i.e slots de l'ordonnancement primaire) ou slots bonus pour transmettre ses messages.

B.5.1.3 Evaluation de performance d'AMSA

• Expériences orientées retransmission:
Nous évaluons le nombre de slots supplémentaires alloués lorsque chaque noeud demande 5%, 10%, 20% de ses slots réguliers comme des slots bonus. Ces pourcentages reflètent le degré de perte de paquets (faible, modéré, élevé).



(a) (b)

Figure B.6: Pourcentage du nombre de slots supplémntaires. (**a**) dans des configurations T_t; (**b**) dans des configurations T_n.

Nous nous focalisons sur des topologies T_t et T_n de 20 noeuds. Comme le montre la Figure B.6: les T_t configurations sont plus gourmandes en termes de slots supplémentaires que les T_n. En plus, dans les configurations de T_n, le nombre de slots supplémentaires attribués est équilibré entre tous les noeuds. Comme illustré sur la figure B.6(b), pour une demande supplémentaire d'au plus 10% des slots réguliers, 80% des noeuds exigent seulement zéro, un, deux ou trois slots supplémentaires. Par conséquent, l'impact des demandes de bonus sur la durée de collecte, dans le cas des retransmissions, est plus conséquent dans les configurations T_t que dans les configurations T_n.

• Expériences orientées changement temporaire des trafics générés par l'application:
L'ordonnancement primaire est donné par MODESA. Nous fixons le pourcentage de noeuds qui exigent un slot supplémentaire (20%). Nous avons réalisé des expériences avec des réseaux

de 10 à 100 noeuds. Tout d'abord, nous avons évalué le nombre moyen de slot supplémentaires nécessaires avec AMSA (c'est à dire les slots créés en plus de l'ordonnancement primaire). Les résultats sont représentés dans la figure B.7

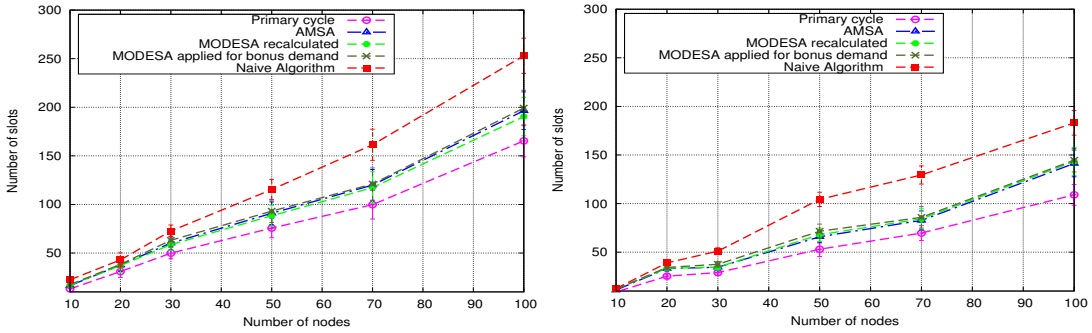


Figure B.7: Nombre moyen de slots supplémentaires lorsque 20% des noeuds demandent un slot bonus (**a**) dans configurations T_t; (**b**) dans configurations T_n.

Nous remarquons que MODESA recalculé (i.e. MODESA exécuté en tenant compte des demandes initiales et des demandes de bonus) et AMSA fournissent des performances très proches quand 5% et 10% de noeuds ont une demande de slot bonus. Cela illustre le mérite d'AMSA qui fournit le même niveau de performance que MODESA recalculé, mais avec moins de complexité. Figure B.7 montre également que MODESA appliqué pour les demandes de bonus uniquement nécessite plus de slots pour ordonnancer une demande supplémentaire que AMSA et MODESA recalculé. Ceci peut être expliqué par le fait que MODESA appliqué pour des les demandes de bonus calcule l'ordonnancement complémentaire séparément du primaire. Seules les demandes de slots bonus sont ordonnancés dans cet ordonnancement complémentaire.

B.6 Allocation distribuée conjointe de slots et canaux pour les applications collecte de données

Les algorithmes centralisés d'allocation conjointe de slot et canaux sont intrinsèquement inefficaces dans les RCSFs à grande échelle. C'est pourquoi, cette section aborde une solution distribué d'ordonnancement dans les RCsF multicanaux.

B.6.1 WAVE: algorithme distribué d'allocation conjointe de slots temporels et canaux pour les applications de collecte de données

WAVE ordonnance les noeuds par vagues successives. Dans chaque vague, chaque noeud ayant un paquet à transmettre obtient un slot temporel et un canal. La première vague constitue un motif. Chaque vague suivante est un sous-ensemble optimisé de la première vague: seuls les slots qui contiendront des transmissions vont se répéter et se produire toujours dans le même ordre que dans la première vague.

La principale différence entre les modes centralisé et distribué de WAVE, concerne les connaissances requises. En mode centralisé, l'entité centrale qui exécute WAVE a la connaissance de toutes les informations relatives à chaque noeud. En mode distribué, tout noeud u ne connaît que les informations relatives à l'ensemble de ses noeuds conflictuels. Pour obtenir ces informations, des messages sont échangés entre les noeuds voisins. En outre, un message spécifique est créé pour informer les noeuds voisins d'une assignation (slot temporel, canal).

En mode distribué, l'assignation du slot temporel et du canal est construite par assemblage de toutes les assignations partielles connues par les noeuds. Plus concrètement, pour calculer la première vague, il suffit pour chaque noeud u de connaître: ses noeuds conflictuels $Conflict(u)$ et le nombre de transmissions $Trans(v)$ de chaque noeud $v \in Conflict(u)$.

Ensuite, tout noeud u envoie à son parent le nombre maximum de transmissions pour chaque slot qu'il connaît. A la fin de la première vague, le puits calcule T, le nombre total de slots dans le motif et pour chaque slot t tel que $1 \leq t \leq T$, le nombre maximal de transmissions $Maxtrans(t)$. Il envoie cette information à tous les noeuds via l'arbre de collecte. Les noeuds sont maintenant en mesure de calculer les prochaines vagues selon la règle R0 ci-dessous:

Rule R0: Tout noeud u ayant le slot j et le canal c dans la première vague, avec $1 \leq j \leq t$, a également le slot $s(k)$ et le canal c dans la vague k, avec $1 \leq k \leq Trans(u)$,

$$s(k) = \sum_{w=1}^{k-1} \sum_{t=1}^{T} \delta_{t,w} + \sum_{t=1}^{j} \delta_{t,k} \tag{B.1}$$

avec $\delta_{t,w} = 1$ si et seulement si $MaxTrans(t) \geq w$ et 0 sinon.

• **Propriétés:**
Sous les hypothèses spécifiées dans la thèse, nous montrons les propriétés suivantes:

Property 7. *Equivalence des modes centralisés et distribués:*
Pour toute topologie considérée, pour toute collecte de données sans agrégation et pour tout trafic de données, le mode centralisé et le mode distribué de WAVE fournissent la même allocation conjointe de slot temporel et de canal.

Property 8. *Efficacité: WAVE, sous les hypothèses proposées dans la section 1, assure que:*

- *si la transmission d'un noeud est prévue dans un slot donné, le noeud a forcément un message à transmettre;*
- *si un paquet est transmis dans un cycle de collecte de données, il atteint le puits dans ce cycle.*

B.6.2 DiSCA: Une version optimisée de WAVE

DiSCA hérite des grands principes de WAVE. Néanmoins, dans DiSCA deux vagues successives peuvent se chevaucher. En effet, DiSCA essaie d'ordonnancer au plus tôt le noeud,

ayant la plus haute priorité parmi ses noeuds conflictuels, c'est à dire dans le plus petit slot temporel où ce noeud a au moins un paquet dans son tampon.

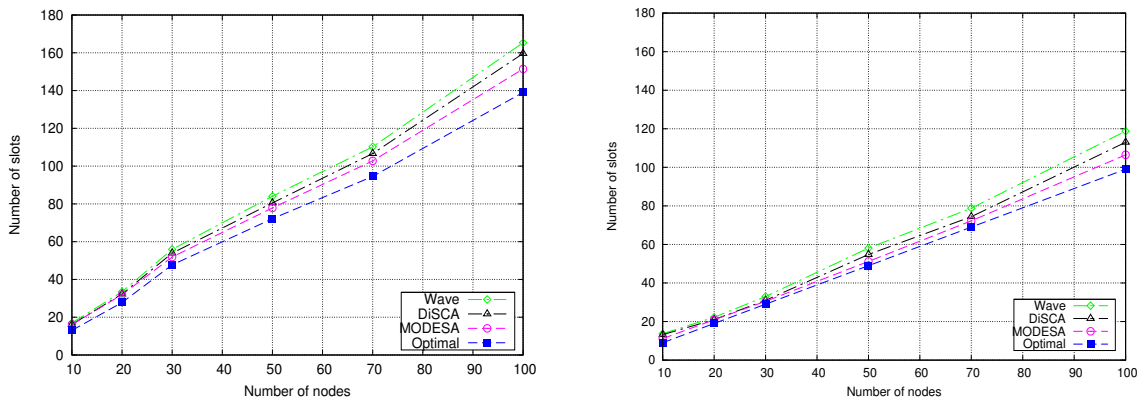
DiSCA se distingue de WAVE par la règle suivante:
Rule R: la i^{ime} transmission de u est pévue dans le premier slot t où u a un paquet à transmettre et sur le premier canal c lorsque les conditions suivantes sont remplies: u a une interface radio disponible, le parent de u a une interface radio disponible et la transmission de u n'est pas incompatible avec la transmission d'un noeud déjà ordonnancée dans le slot t et sur le canal c.

B.6.3 Evaluation comparative de WAVE et DiSCA

B.6.3.1 Comparaison de WAVE et DiSCA avec un ordonnancement optimal et MODESA

En supposant que tout noeud génère un paquet et 2 canaux sont disponibles sur chaque noeud, nous comparons le nombre de slot obtenus par l'ordonnancement optimal, MODESA, WAVE et DiSCA.

Pour des configurations de 100 noeuds, WAVE est à moins de 18% (respectivement 17%) de l'optimal dans les configurations T_t (respectivement configurations T_n). En outre, WAVE est un peu moins performant que DiSCA mais il est beaucoup plus simple. MODESA est à moins de 11% (respectivement 10%) loin de l'optimal dans les configurations T_t (respectivement configurations T_n) comme le montre la figure B.8



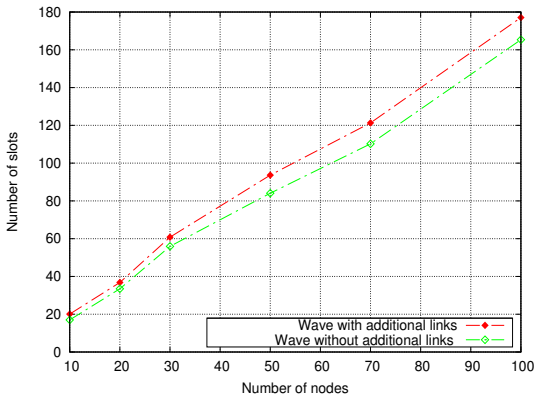
(a) in configurations T_t (b) dans configurations T_n

Figure B.8: Comparaison du nombre de slots produit par $WAVE$, l'ordonnancement optimal et MODESA: trafic homogène

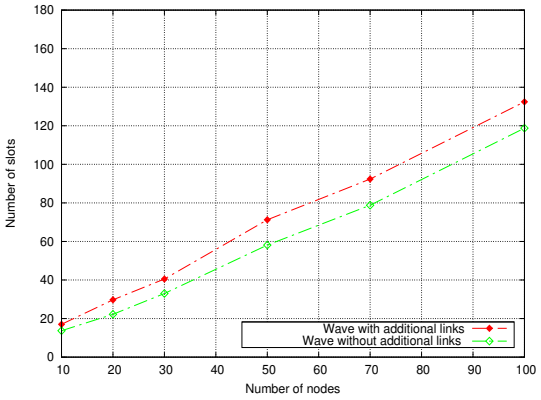
B.6.3.2 L'impact de l'existence d'autres liens interférents sur l'ordonnancement obtenu par WAVE

WAVE ne nécessite pas que tous les liens interférents soient éliminés. Il prend facilement en compte la présence de liens topologiques supplémentaires à ceux existant dans l'arbre de

collecte. En effet, l'ensemble de conflit d'un noeud est une boîte noire pour l'algorithme.



(a) configurations T_t



(b) configurations T_n

Figure B.9: Nombre de slots produit par $WAVE$ avec des liens interférents additionels

L'existence de liens topologiques qui n'existent pas dans l'arbre de routage induit d'avantage de conflits. Par conséquent, les possibilités de transmissions parallèles sont réduites conduisant à un plus grand nombre de slots: pour 100 noeuds, les liens ajoutés aboutissent à 8% (respectivement 11%) slots supplémentaires pour WAVE dans les configurations T_t (respectivement dans les configurations T_n) (voir figure B.9). Le fait que cette augmentation reste faible est dû à notre définition très précise des noeuds conflictuels.

B.7 Conclusion et perspectives

Cette thèse s'est particulièrement intéressée aux applications de collecte de données sans agrégation. Dans de telles applications, les capteurs mesurent des paramètres de l'environnement et envoient leurs données, directement ou en multi-sauts, au noeud puits qui les traite. Ce modèle de communication est utilisé dans la plupart des applications industrielles supportées par les réseaux de capteurs sans fil. Pour réduire les délais de collecte de bout-en-bout, les données doivent être remises aussi vite que possible au puits. Les caractéristiques intrinsèques des réseaux de capteurs telles que le fonctionnement sur batterie couplé avec les problèmes de contention d'accès au médium et les interférences, soulèvent de grands défis. Pour augmenter le débit et améliorer la robustesse des communications, nous utilisons le multicanal.

Ainsi nous nous sommes intéressés à l'assignation de slots temporels et de canaux pour les applications de collecte de données sans agrégation. Les noeuds proches du puits demandent plus de bande passante que les noeuds feuilles dans l'arbre de collecte. L'algorithme d'assignation de slots temporels et de canaux doit minimiser le nombre de slots tout en assurant la remontée des données générées par les capteurs en un seul cycle. Ceci a pour avantage de réduire les délais de collecte et assurer une meilleure cohérence temporelle des données collectées. Nous avons déterminé les bornes inférieures du nombre de slots nécessaires à la collecte de données en environnement multicanal lorsque le puits est équipé de plusieurs interfaces radio et les noeuds ont des demandes hétérogènes.

Nous avons montré que les configurations équilibrées en termes de trafic entre les différents sous-arbres, sont moins gourmandes en slots que les configurations, où un sous-arbre impose le nombre de slots nécessaires à la collecte.

Dans des environnements statiques où les demandes des applications sont connues à l'avance et l'environnement sans fil est partiellement connu aussi, nous avons proposé MODESA une solution centralisée d'allocation conjointe de slots temporels et canaux. Selon les simulations, MODESA permet d'atteindre des performances proches de l'ordonnancement optimal. MODESA a été étendu pour les applications de collecte de données qui utilisent plusieurs puits pour la collecte de données. Ainsi, nous avons proposé MUSIKA qui présente une solution originale d'ordonnancement sans conflit d'arbres de collecte de données pouvant se recouvrir partiellement dans les réseaux de capteurs multicanaux.

Nous avons ensuite adopté MODESA au routage multichemin: en équilibrant mieux le trafic transmis vers le puits entre les différents chemins, MODESA permet d'optimiser les délais de collecte.

En outre, les retransmissions ou les changements soudains des trafics soumis nécessitent des changements dans l'allocation des slots et canaux. Ainsi, la génération d'un nouvel ordonnancement pour l'ensemble du réseau (en particulier dans les RCsF à grande échelle), est une tâche gourmande en ressources. C'est pourquoi nous avons proposé un mécanisme adaptatif incrémental d'allocation conjointe de slots et canaux appelé AMSA. Les simulations indiquent que AMSA assure une collecte de données efficace en particulier pour des transmissions supplémentaires faibles (5%) et modérées (10%).

Dans une dernière contribution, nous avons proposé, WAVE, une solution distribuée pour allocation conjointe de slots et canaux basée sur le principe des vagues successives. Les simulations ont confirmé l'efficacité de WAVE et ont démontré le bon compromis entre simplicité de la solution et rapidité de la collecte de données.

Parmi les perspectives envisagées pour ces travaux, nous pouvons citer:

- Evaluation des performances de nos algorithmes dans des réseaux réels de capteurs sans fil évoluant dans des conditions proches de l'environnement opérationnel.
- Adaptation de nos solutions aux deux types de collecte de données: avec agrégation, hybride (ex.: avec agrégation intra-cluster et sans agrégation inter-cluster).
- Les réseaux radios cognitifs offrent l'accès à un nouveau spectre qui doit être utilisé d'une façon opportuniste. Comment les protocoles dans cette thèse peuvent-ils être étendues à ce type de réseau?.

Bibliography

[802.15.4] 802.15.4. *http://www.ieee802.org/15/pub/TG4.htm.* 33

[Accettura 2013] Nicola Accettura, Maria Rita Palattella, Gennaro Boggia, Luigi Alfredo Grieco and Mischa Dohler. *Decentralized Traffic Aware Scheduling for multi-hop Low power Lossy Networks in the Internet of Things.* In WOWMOM, pages 1–6. IEEE, 2013. 137

[Akan 2009] Ozgur B. Akan, Osman B. Karli and Ozgur Ergul. *Cognitive radio sensor networks.* Netwrk. Mag. of Global Internetwkg., vol. 23, no. 4, pages 34–40, July 2009. 163

[Akkaya 2005] Kemal Akkaya and Mohamed F. Younis. *A survey on routing protocols for wireless sensor networks.* Ad Hoc Networks, vol. 3, no. 3, pages 325–349, 2005. 12, 14, 15, 16

[Al-Ayyoub 2010] M. Al-Ayyoub and H. Gupta. *Joint Routing, Channel Assignment, and Scheduling for Throughput Maximization in General Interference Models.* Mobile Computing, IEEE Transactions on, vol. 9, no. 4, pages 553–565, 2010. 30

[Alicherry 2006a] M. Alicherry, R. Bhatia and Li Erran Li. *Joint Channel Assignment and Routing for Throughput Optimization in Multiradio Wireless Mesh Networks.* Selected Areas in Communications, IEEE Journal on, vol. 24, no. 11, pages 1960–1971, 2006. 24

[Alicherry 2006b] M. Alicherry, R. Bhatia and Li Erran Li. *Joint Channel Assignment and Routing for Throughput Optimization in Multiradio Wireless Mesh Networks.* Selected Areas in Communications, IEEE Journal on, vol. 24, no. 11, pages 1960–1971, 2006. 26

[Anastasi 2009] Giuseppe Anastasi, Marco Conti, Mario Di Francesco and Andrea Passarella. *Energy conservation in wireless sensor networks: A survey.* Ad Hoc Networks, vol. 7, no. 3, pages 537–568, 2009. 12

[Baccour 2012] Nouha Baccour, Anis Koubâa, Luca Mottola, Marco Antonio Zúñiga, Habib Youssef, Carlo Alberto Boano and Mário Alves. *Radio Link Quality Estimation in*

Wireless Sensor Networks: A Survey. ACM Trans. Sen. Netw., vol. 8, no. 4, pages 34:1–34:33, September 2012. 22, 29

[BelAirNetworks 2007] BelAirNetworks. *BelAirNetworks, Bulding inclusive municipal wireless mesh networks.* Rapport technique, BelAirNetworks, 2007. 25

[Bhardwaj 2002] M. Bhardwaj and A.P. Chandrakasan. *Bounding the lifetime of sensor networks via optimal role assignments.* In INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 3, pages 1587–1596 vol.3, 2002. 15

[Blass 2008] E.-O. Blass, J. Horneber and M. Zitterbart. *Analyzing Data Prediction in Wireless Sensor Networks.* In Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE, pages 86–87, 2008. 14

[Blough 2002] Douglas M. Blough and Paolo Santi. *Investigating Upper Bounds on Network Lifetime Extension for Cell-based Energy Conservation Techniques in Stationary Ad Hoc Networks.* In Proceedings of the 8th Annual International Conference on Mobile Computing and Networking, MobiCom '02, pages 183–192, New York, NY, USA, 2002. ACM. 10

[Bluetooth] Bluetooth. *http://www.ieee802.org/15/.* 36

[Borms 2010] Joris Borms, Kris Steenhaut and Bart Lemmens. *Low-Overhead Dynamic Multi-channel MAC for Wireless Sensor Networks.* In Jorge Sá Silva, Bhaskar Krishnamachari and Fernando Boavida, editeurs, EWSN, volume 5970 of *Lecture Notes in Computer Science*, pages 81–96. Springer, 2010. 29, 34, 37, 41

[Cardei 2005a] M. Cardei, M.T. Thai, Yingshu Li and Weili Wu. *Energy-efficient target coverage in wireless sensor networks.* In INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, volume 3, pages 1976–1984 vol. 3, 2005. 16

[Cardei 2005b] Mihaela Cardei and Ding-Zhu Du. *Improving Wireless Sensor Network Lifetime Through Power Aware Organization.* Wirel. Netw., vol. 11, no. 3, pages 333–340, May 2005. 16

[Chakrabarty 2002] K. Chakrabarty, S.S. Iyengar, Hairong Qi and Eungchun Cho. *Grid coverage for surveillance and target location in distributed sensor networks.* Computers, IEEE Transactions on, vol. 51, no. 12, pages 1448–1453, 2002. 16

[Chen 2002] Benjie Chen, Kyle Jamieson, Hari Balakrishnan and Robert Morris. *Span: An Energy-efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks.* Wirel. Netw., vol. 8, no. 5, pages 481–494, September 2002. 16

[Chen 2009] Sixia Chen, Matthew Coolbeth, Hieu Dinh, Yoo-Ah Kim and Bing Wang. *Data Collection with Multiple Sinks in Wireless Sensor Networks.* In Proceedings of the 4th International Conference on Wireless Algorithms, Systems, and Applications, WASA '09, pages 284–294, Berlin, Heidelberg, 2009. Springer-Verlag. 94

[Chen 2011] J. Chen, Q. Yu, P. Cheng, Yo. Sun, Y. Fan and X. Shen. *Game Theoretical Approach for Channel Allocation in Wireless Sensor and Actuator Networks.* IEEE Transactions in Automatic Control, October 2011. 162

[Chiasserini 2002] Carla-Fabiana Chiasserini, Imrich Chlamtac, Paolo Monti and Antonio Nucci. *Energy Efficient Design of Wireless Ad Hoc Networks.* In Proceedings of the Second International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; and Mobile and Wireless Communications, NETWORKING '02, pages 376–386, London, UK, UK, 2002. Springer-Verlag. 10

[Crichigno 2008] Jorge Crichigno, Min-You Wu and Wei Shu. *Protocols and architectures for channel assignment in wireless mesh networks.* Ad Hoc Networks, vol. 6, no. 7, pages 1051–1077, 2008. 35

[Dai 2004] Fei Dai and Jie Wu. *An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks.* IEEE Transactions on Parallel and Distributed Systems, vol. 15, no. 10, pages 908–920, 2004. 12

[Das 2006] S.M. Das, H. Pucha, D. Koutsonikolas, Y.C. Hu and D. Peroulis. *DMesh: Incorporating Practical Directional Antennas in Multichannel Wireless Mesh Networks.* Selected Areas in Communications, IEEE Journal on, vol. 24, no. 11, pages 2028–2039, 2006. 24

[Dietrich 2009] Isabel Dietrich and Falko Dressler. *On the Lifetime of Wireless Sensor Networks.* ACM Trans. Sen. Netw., vol. 5, no. 1, pages 5:1–5:39, February 2009. 10

[Durmaz Incel 2008] O. Durmaz Incel, A. Ghosh, B. Krishnamachari and K. Chintalapudi. *Multi-Channel Scheduling for Fast Convergecast in Wireless Sensor Networks*, September 2008. 136, 154

[Fan 2006a] Kai-Wei Fan, Sha Liu and Prasun Sinha. *On the Potential of Structure-Free Data Aggregation in Sensor Networks.* In INFOCOM. IEEE, 2006. 12

[Fan 2006b] Kai-Wei Fan, Sha Liu and Prasun Sinha. *Scalable data aggregation for dynamic events in sensor networks.* In Andrew T. Campbell, Philippe Bonnet and John S. Heidemann, editeurs, SenSys, pages 181–194. ACM, 2006. 12

[Fotue 2012] D. Fotue, H. Labiod and T. Engel. *Performance Evaluation of Hybrid Channel Assignment for Wireless Sensor Networks.* In Mobile Ad-hoc and Sensor Networks (MSN), 2012 Eighth International Conference on, pages 31–37, 2012. 34

[Galbreath 2006a] J. Galbreath and J. Frolik. *Channel allocation strategies for wireless sensors statically deployed in multipath environments.* In Information Processing in Sensor Networks, 2006. IPSN 2006. The Fifth International Conference on, pages 334–341, 2006. 32

[Galbreath 2006b] J. Galbreath and J. Frolik. *Channel allocation strategies for wireless sensors statically deployed in multipath environments.* In Information Processing in Sensor Networks, 2006. IPSN 2006. The Fifth International Conference on, pages 334–341, 2006. 33

[Gandham 2008] Shashidhar Gandham, Ying Zhang and Qingfeng Huang. *Distributed Time-optimal Scheduling for Convergecast in Wireless Sensor Networks.* Comput. Netw., vol. 52, no. 3, pages 610–629, February 2008. 136

[Garcia-Luna-Aceves 2000] Asimakis Tzamaloukas Garcia-Luna-Aceves and J. J. Garcia luna aceves. *Channel Hopping Multiple Access with Packet Trains for Ad Hoc Networks.* In In In Proc. IEEE Mobile Multimedia Communications (MoMuC â00, 2000. 36

[Gedik 2007] Bugra Gedik, Ling Liu and Philip S. Yu. *ASAP: An Adaptive Sampling Approach to Data Collection in Sensor Networks.* IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 12, pages 1766–1783, 2007. 12

[Ghosh 2009] A. Ghosh, O.D. Incel, V.S.A. Kumar and B. Krishnamachari. *Multi-channel scheduling algorithms for fast aggregated convergecast in sensor networks.* In Mobile Adhoc and Sensor Systems, 2009. MASS '09. IEEE 6th International Conference on, pages 363–372, 2009. 31, 82, 154

[GLPK 2011] GLPK. *http://www.gnu.org/software/glpk/*, 2011. 69, 101

[Gobriel 2009a] S. Gobriel, D. Mosse and R. Cleric. *TDMA-ASAP: Sensor Network TDMA Scheduling with Adaptive Slot-Stealing and Parallelism.* In Distributed Computing Systems, 2009. ICDCS '09. 29th IEEE International Conference on, pages 458–465, 2009. 17

[Gobriel 2009b] S. Gobriel, D. Mousse and R. Cleric. *TDMA-ASAP: sensor network TDMA scheduling with adaptive slot stealing and parallelism.* In ICDCS, 2009. 109

[Goel 2001] Samir Goel and Tomasz Imielinski. *Prediction-based Monitoring in Sensor Networks: Taking Lessons from MPEG.* SIGCOMM Comput. Commun. Rev., vol. 31, no. 5, pages 82–98, October 2001. 12

[Goel 2006] S. Goel, A. Passarella and T. Imielinski. *Using buddies to live longer in a boring world [sensor network protocol].* In Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on, pages 5 pp.–346, 2006. 12

[Gonga 2012] A. Gonga, O. Landsiedel, P. Soldati and M. Johansson. *Revisiting Multi-channel Communication to Mitigate Interference and Link Dynamics in Wireless Sensor Networks.* In Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on, pages 186–193, 2012. 33, 36, 39

[Han 2009] Bo Han, V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy 0002 and Anand Srinivasan. *Distributed Strategies for Channel Allocation and Scheduling*

in Software-Defined Radio Networks. In INFOCOM, pages 1521–1529. IEEE, 2009. 137

[Heinzelman 2000] 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. 12

[Heinzelman 2002] W.B. Heinzelman, A.P. Chandrakasan and H. Balakrishnan. *An application-specific protocol architecture for wireless microsensor networks.* Wireless Communications, IEEE Transactions on, vol. 1, no. 4, pages 660–670, 2002. 12

[Hiromori 2012] A. Hiromori, A. Uchiyama, H. Yamaguchi and T. Higashino. *Deadline-Aware Data Collection in CSMA/CA-based Multi-Sink Wireless Sensor Networks.* In Proceedings of the Sixth International Conference on Mobile Computing and Ubiquitous Networking, ICMU '12, pages 284–294, 2012. 94

[Incel 2011] Ozlem Durmaz Incel. *A survey on multi-channel communication in wireless sensor networks.* Computer Networks, vol. 55, no. 13, pages 3081–3099, 2011. 21

[Incel 2012] Ozlem Durmaz Incel, Amitabha Ghosh, Bhaskar Krishnamachari and Krishnakant Chintalapudi. *Fast Data Collection in Tree-Based Wireless Sensor Networks.* IEEE Transactions on Mobile Computing, vol. 11, no. 1, pages 86–99, 2012. 37, 42, 51, 67, 82, 87, 133, 160

[Ingelrest 2005] F. Ingelrest and D. Simplot-Ryl. *Maximizing the probability of delivery of multipoint relay broadcast protocol in wireless ad hoc networks with a realistic physical layer.* Rapport technique, INRIA, 2005. xi, 29

[Ingelrest 2007] François Ingelrest, David Simplot-Ryl and Ivan Stojmenovic. *Smaller Connected Dominating Sets in Ad Hoc and Sensor Networks based on Coverage by Two-Hop Neighbors.* In COMSWARE. IEEE, 2007. 12

[Jain 2004] Ankur Jain and Edward Y. Chang. *Adaptive Sampling for Sensor Networks.* In Proceeedings of the 1st International Workshop on Data Management for Sensor Networks: In Conjunction with VLDB 2004, DMSN '04, pages 10–16, New York, NY, USA, 2004. ACM. 12

[Kanzaki 2009] Akimitsu Kanzaki, Takahiro Hara and Shojiro Nishio. *On a TDMA Slot Assignment Considering the Amount of Traffic in Wireless Sensor Networks.* In AINA Workshops, pages 984–989. IEEE Computer Society, 2009. 109

[Kim 2008] Youngmin Kim, Hyojeong Shin and Hojung Cha. *Y-MAC: An Energy-Efficient Multi-channel MAC Protocol for Dense Wireless Sensor Networks.* In Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on, pages 53–63, 2008. 29, 34, 37, 41, 67

[Kimura 2005] N. Kimura and S. Latifi. *A survey on data compression in wireless sensor networks.* In Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on, volume 2, pages 8–13 Vol. 2, 2005. 12

[Ko 2007] Bong-Jun Ko, V. Misra, J. Padhye and D. Rubenstein. *Distributed Channel Assignment in Multi-Radio 802.11 Mesh Networks.* In Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE, pages 3978–3983, 2007. 25

[Kodialam 2005] Murali Kodialam and Thyaga Nandagopal. *Characterizing the Capacity Region in Multi-radio Multi-channel Wireless Mesh Networks.* In Proceedings of the 11th Annual International Conference on Mobile Computing and Networking, MobiCom '05, pages 73–87, New York, NY, USA, 2005. ACM. 24

[Kumar 2005] S. Kumar, A. Arora and T.H. Lai. *On the lifetime analysis of always-on wireless sensor network applications.* In Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on, pages 3 pp.–188, 2005. 10

[Kyasanur 2006] Pradeep Kyasanur and Nitin H. Vaidya. *Routing and Link-layer Protocols for Multi-channel Multi-interface Ad Hoc Wireless Networks.* SIGMOBILE Mob. Comput. Commun. Rev., vol. 10, no. 1, pages 31–43, January 2006. 25, 34

[Lee 2008] W.L. Lee, A. Datta and R. Cardell-Oliver. *FlexiTP: A Flexible-Schedule-Based TDMA Protocol for Fault-Tolerant and Energy-Efficient Wireless Sensor Networks.* Parallel and Distributed Systems, IEEE Transactions on, vol. 19, no. 6, pages 851–864, 2008. 17

[Li 2008] Cheng Li, Pu Wang, Hsiao-Hwa Chen and M. Guizani. *A Cluster Based On-demand Multi-Channel MAC Protocol for Wireless Multimedia Sensor Networks.* In Communications, 2008. ICC '08. IEEE International Conference on, pages 2371–2376, 2008. 27, 34, 41

[Li 2010] Jinbao Li, Desheng Zhang, Longjiang Guo, Shouling Ji and Yingshu Li. *ARM: An asynchronous receiver-initiated multichannel MAC protocol with duty cycling for WSNs.* In Performance Computing and Communications Conference (IPCCC), 2010 IEEE 29th International, pages 114–121, 2010. 34, 35, 37

[Li 2011a] Jinbao Li, Xiaohang Guo and Longjiang Guo. *Joint routing, scheduling and channel assignment in multi-power multi-radio wireless sensor networks.* IEEE International Performance Computing and Communications Conference, vol. 0, pages 1–8, 2011. 38, 41

[Li 2011b] Jinbao Li, Xiaohang Guo and Longjiang Guo. *Joint routing, scheduling and channel assignment in multi-power multi-radio wireless sensor networks.* In Sheng Zhong, Dejing Dou and Yu Wang 0003, editeurs, IPCCC, pages 1–8. IEEE, 2011. 86

[Liang 2010] Chieh-Jan Mike Liang, Nissanka Bodhi Priyantha, Jie Liu and Andreas Terzis. *Surviving Wi-fi Interference in Low Power ZigBee Networks.* In Proceedings of the

8th ACM Conference on Embedded Networked Sensor Systems, SenSys '10, pages 309–322, New York, NY, USA, 2010. ACM. 21

[Lin 2011] Chih-Kuang Lin, Vladimir I. Zadorozhny, Prashant V. Krishnamurthy, Ho-Hyun Park and Chan-Gun Lee. *A Distributed and Scalable Time Slot Allocation Protocol for Wireless Sensor Networks.* IEEE Transactions on Mobile Computing, vol. 10, no. 4, pages 505–518, 2011. 136

[Lu 2004] G. Lu, B. Krishnamachari and C.S. Raghavendra. *An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks.* In Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, pages 224– , 2004. 17

[Luo 2005] Jun Luo and J-P Hubaux. *Joint mobility and routing for lifetime elongation in wireless sensor networks.* In INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, volume 3, pages 1735–1746 vol. 3, 2005. 15

[Mabrouk 2014] Ons Mabrouk, Hanen Idoudi, Ichrak Amdouni, Ridha Soua, Pascale Minet and Leila Saidane. *OTICOR: Opportunistic Time Slot Assignment in Cognitive Radio Sensor Networks.* In AINA. IEEE, 2014. 163

[Macedo 2009] Mário Macedo, António Grilo and Mário Serafim Nunes. *Distributed Latency-Energy Minimization and interference avoidance in TDMA Wireless Sensor Networks.* Computer Networks, vol. 53, no. 5, pages 569–582, 2009. 94

[Mahfoudh 2008] Saoucene Mahfoudh and Pascale Minet. *An Energy Efficient Routing Based on OLSR in Wireless Ad Hoc and Sensor Networks.* In Proceedings of the 22Nd International Conference on Advanced Information Networking and Applications - Workshops, AINAW '08, pages 1253–1259, Washington, DC, USA, 2008. IEEE Computer Society. 12

[Mahfoudh 2010a] S. Mahfoudh, P. Minet and I. Amdouni. *Energy Efficient Routing and Node Activity Scheduling in the OCARI Wireless Sensor Network.* Journal of Future Internet, 2010. 36

[Mahfoudh 2010b] Saoucene Mahfoudh, Gerard Chalhoub, Pascale Minet, Michel Misson and Ichrak Amdouni. *Node Coloring and Color Conflict Detection in Wireless Sensor Networks.* Future Internet, vol. 2, no. 4, pages 469–504, 2010. 17

[Marbini 2003] A Djafari Marbini and L. E. Sacks. *Adaptive Sampling Mechanisms in Sensor Networks.* In LCS, 2003. 12

[Marina 2005] M.K. Marina and S.R. Das. *A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks.* In Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on, pages 381–390 Vol. 1, 2005. 24, 25

[Meguerdichian 2003] S. Meguerdichian and M. Potkonjak. *Low power 0/1 coverage and scheduling techniques in sensor networks.* Rapport technique, UCLA, 2003. 16

[Miao 2008] Cui Miao, Yin Junxun, Nie JianYao, Zhang JinJuan and Cao YingLie. *An evolutionary dynamic slots assignment algorithm based on P-TDMA for mobile ad hoc networks.* In Communication Systems, 2008. ICCS 2008. 11th IEEE Singapore International Conference on, pages 579–582, 2008. 108

[Minet 2009] P. Minet. Energy efficient routing, page xx. Bentham Science, 2009. 10, 11

[Mottola 2011] Luca Mottola and Gian Pietro Picco. *MUSTER: Adaptive Energy-Aware Multisink Routing in Wireless Sensor Networks.* IEEE Transactions on Mobile Computing, vol. 10, no. 12, pages 1694–1709, 2011. 95

[Naveed 2007a] A. Naveed, S.S. Kanhere and S.K. Jha. *Topology Control and Channel Assignment in Multi-Radio Multi-Channel Wireless Mesh Networks.* In Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on, pages 1–9, 2007. 24

[Naveed 2007b] A. Naveed, S.S. Kanhere and S.K. Jha. *Topology Control and Channel Assignment in Multi-Radio Multi-Channel Wireless Mesh Networks.* In Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on, pages 1–9, 2007. 25

[Octave] Octave. *http://www.gnu.org/software/octave/.* 75, 102, 121, 153

[Ortiz 2010] Jorge Ortiz and David E. Culler. *Multichannel reliability assessment in real world WSNs.* In Tarek F. Abdelzaher, Thiemo Voigt and Adam Wolisz, editeurs, IPSN, pages 162–173. ACM, 2010. 39

[Ou 2007] Chia-Ho Ou and Kuo-Feng Ssu. *Routing with Mobile Relays in Opportunistic Sensor Networks.* In Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on, pages 1–5, 2007. 15

[Palattella 2012] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco and G. Boggia. *Traffic Aware Scheduling Algorithm for Reliable Low-Power Multi-Hop IEEE 802.15.4e Networks.* In Proc. of IEEE Int. Symp. on Personal, Indoor and Mobile Radio Commun., PIMRC, Sydney, Australia, Sep. 2012. 67, 137

[Papadimitriou 1978] Christos H. Papadimitriou. *The complexity of the capacitated tree problem.* Networks, vol. 8, no. 3, pages 217–230, 1978. 59

[Papadimitriou 2006] I. Papadimitriou and L. Georgiadis. *Energy-aware Routing to Maximize Lifetime in Wireless Sensor Networks with Mobile Sink.* Journal of Communications Software and Systems, vol. 2, 2006. 15

[Pei 2000] Guangyu Pei, Mario Gerla and Tsu-Wei Chen. *Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks.* In ICC (1), pages 70–74, 2000. 12

[Pelusi 2006] L. Pelusi, A. Passarella and M. Conti. *Opportunistic networking: data forwarding in disconnected mobile ad hoc networks.* IEEE Communications Magazine, vol. 44, no. 11, pages 134–141, November 2006. 15

[Phung 2012] Kieu-Ha Phung, Bart Lemmens, Mihail Mihaylov, Dario Di Zenobio, Kris Steenhaut and Lan Tran. *Multi-agent learning for multi-channel wireless sensor networks.* In ICC, pages 6448–6452. IEEE, 2012. 162

[Phung 2013] K-H. Phung, B. lemmens lemmens lemmens lemmens, M. Mihaylov, D. D. Zenobio, K. Steenhaut and L. Tran. *Adaptive Learning Based Scheduling in MultiChannel Protocol for Energy-Efficient Data-Gathering Wireless Sensor Networks.* International Journal of Distributed Sensor Networks, 2013. 162

[Polastre 2004] Joseph Polastre, Jason Hill and David Culler. *Versatile Low Power Media Access for Wireless Sensor Networks.* In Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems, SenSys '04, pages 95–107, New York, NY, USA, 2004. ACM. 17

[Quin 2003] L. Quin and T. Kunz. *On-demand routing in MANETs: The impact of a realistic physical layer model.* In the International Conference on Ad-Hoc, Mobile, and Wireless Networks, 2003. 29

[Rajendran 2003] V. Rajendran, K. Obraczka and J. j. Garicia-Luna-Aceves. *Energy-Efficient, Colision-free Medium Access Control for Wireless Sensor Networks.* In ACM SenSys, 2003. 17

[Rajendran 2005] V. Rajendran, J.J. Garcia-Luna-Aveces and K. Obraczka. *Energy-efficient, application-aware medium access for sensor networks.* In Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on, pages 8 pp.–630, 2005. 17

[Raman 2009] V. Raman and N. H. Vaidya. *Adjacent Channel Interference Reduction in Multichannel Wireless Networks Using Intelligent Channel Allocation.* Rapport technique, University of Illinois, September 2009. 21

[Raman 2010] Bhaskaran Raman, Kameswari Chebrolu, Sagar Bijwe and Vijay Gabale. *PIP: a connection-oriented, multi-hop, multi-channel TDMA-based MAC for high throughput bulk transfer.* In Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, SenSys '10, pages 15–28, New York, NY, USA, 2010. ACM. 67

[Raniwala 2004] Ashish Raniwala, Kartik Gopalan and Tzi cker Chiueh. *Centralized Channel Assignment and Routing Algorithms for Multi-channel Wireless Mesh Networks.* SIGMOBILE Mob. Comput. Commun. Rev., vol. 8, no. 2, pages 50–65, April 2004. 24

[Raniwala 2005a] A. Raniwala and T. Chiueh. *Architecture and algorithm for an ieee 802.11-based multi-channel wireless mesh network.* In IEEE INFOCOM, 2005. 25

[Raniwala 2005b] A. Raniwala and Tzi cker Chiueh. *Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network.* In INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, volume 3, pages 2223–2234 vol. 3, 2005. 24, 25

[Rhee 2008] Injong Rhee, A. Warrier, M. Aia, Jeongki Min and M.L. Sichitiu. *Z-MAC: A Hybrid MAC for Wireless Sensor Networks.* Networking, IEEE/ACM Transactions on, vol. 16, no. 3, pages 511–524, 2008. 17

[Ruzzelli 2006] A.G. Ruzzelli, G. O'Hare, R. Jurdak and R. Tynan. *Advantages of Dual Channel MAC for Wireless Sensor Networks.* In Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on, pages 1–3, 2006. 26

[Sha 2011a] Mo Sha, Gregory Hackmann and Chenyang Lu. *ARCH: Practical Channel Hopping for Reliable Home-Area Sensor Networks.* In IEEE Real-Time and Embedded Technology and Applications Symposium, pages 305–315. IEEE Computer Society, 2011. 42

[Sha 2011b] Mo Sha, Gregory Hackmann and Chenyang Lu. *Multi-channel reliability and spectrum usage in real homes: empirical studies for home-area sensor networks.* In Proceedings of the Nineteenth International Workshop on Quality of Service, IWQoS '11, pages 39:1–39:9, Piscataway, NJ, USA, 2011. IEEE Press. 42

[Shah 2003] R.C. Shah, S. Roy, S. Jain and W. Brunette. *Data MULEs: modeling a three-tier architecture for sparse sensor networks.* In Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on, pages 30–41, 2003. 15

[Shin 2012] Dong-Hoon Shin, S. Bagchi and Chih-Chun Wang. *Distributed online channel assignment toward optimal monitoring in multi-channel wireless networks.* In INFOCOM, 2012 Proceedings IEEE, pages 2626–2630, 2012. 26

[Si 2010] Weisheng Si, Selvadurai Selvakennedy and Albert Y. Zomaya. *An overview of Channel Assignment methods for multi-radio multi-channel wireless mesh networks.* Journal of Parallel and Distributed Computing, vol. 70, no. 5, pages 505–524, 2010. 23, 24, 25

[Slimane 2011] Jamila Ben Slimane, Yeqiong Song, Anis Koubaa and Mounir Frikha. *Joint Duty Cycle Scheduling, Resource Allocation and Multi-constrained QoS Routing Algorithm.* In Hannes Frey, Xu Li and Stefan Rührup, editeurs, ADHOC-NOW, volume 6811 of *Lecture Notes in Computer Science*, pages 29–43. Springer, 2011. 40

[So 2004a] J. So and N. Vaidya. *Multi-channel MAC for ad-hoc networks: handling multi-channel hidden terminal using a single transceiver.* In MobiHoc, 2004. 35

[So 2004b] Jungmin So and Nitin H. Vaidya. *Multi-channel Mac for Ad Hoc Networks: Handling Multi-channel Hidden Terminals Using a Single Transceiver.* In Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc '04, pages 222–233, New York, NY, USA, 2004. ACM. 20

[Solutions] Nortel Wireless Solutions. *http://www.nortel.com.* 25

[Song 2008] Jianping Song, Song Han, A.K. Mok, Deji Chen, M. Lucas and M. Nixon. *WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control*. In Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS '08. IEEE, pages 377–386, 2008. 29, 37, 46, 66

[Soro 2005] S. Soro and W.B. Heinzelman. *Prolonging the lifetime of wireless sensor networks via unequal clustering*. In Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International, pages 8 pp.–, 2005. 10

[Stabellini 2009] L. Stabellini. *Design of Reliable Communication Solutions for Wireless Sensor Networks: Managing Interference in Unlicensed Bands*. PhD thesis, KTH, School of Information and Communication Technology (ICT), 2009. 33

[Subramanian 2008a] A.P. Subramanian, H. Gupta, S.R. Das and Jing Cao. *Minimum Interference Channel Assignment in Multiradio Wireless Mesh Networks*. Mobile Computing, IEEE Transactions on, vol. 7, no. 12, pages 1459–1473, 2008. 24

[Subramanian 2008b] A.P. Subramanian, H. Gupta, S.R. Das and Jing Cao. *Minimum Interference Channel Assignment in Multiradio Wireless Mesh Networks*. Mobile Computing, IEEE Transactions on, vol. 7, no. 12, pages 1459–1473, 2008. 26

[Tan 2008] Hwee-Xian Tan, Mun Choon Chan, Peng Yong Kong and Chen-Khong Tham. *A Resource Allocation Scheme for TH-UWB Networks with Multiple Sinks*. In WCNC, pages 1547–1552. IEEE, 2008. 95

[Tang 2005] Jian Tang, Guoliang Xue and Weiyi Zhang. *Interference-aware Topology Control and QoS Routing in Multi-channel Wireless Mesh Networks*. In Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc '05, pages 68–77, New York, NY, USA, 2005. ACM. 24, 25

[Tang 2011] Lei Tang, Yanjun Sun, Omer Gurewitz and David B. Johnson. *EM-MAC: A Dynamic Multichannel Energy-efficient MAC Protocol for Wireless Sensor Networks*. In Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc '11, pages 23:1–23:11, New York, NY, USA, 2011. ACM. 29, 34, 35, 36, 39, 41, 42

[TG4e 2012] TG4e. *http://www.ieee802.org/15/pub/TG4e.html*, 2012. 2, 66, 136, 157, 160

[Tian 2002] Di Tian and Nicolas D. Georganas. *A Coverage-preserving Node Scheduling Scheme for Large Wireless Sensor Networks*. In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, WSNA '02, pages 32–41, New York, NY, USA, 2002. ACM. 10, 11

[Tropos] Tropos. *http://www.tropos.com*. 25

[Tselishchev 2011] Yuriy Tselishchev, Lavy Libman and Athanassios Boulis. *Energy-efficient retransmission strategies under variable TDMA scheduling in body area networks*. In Chun Tung Chou, Tom Pfeifer and Anura P. Jayasumana, editeurs, LCN, pages 374–381. IEEE, 2011. 109

[Tzamaloukas 2001] A. Tzamaloukas and J.J. Garcia-Luna-Aceves. *A receiver-initiated collision-avoidance protocol for multi-channel networks.* In INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 1, pages 189–198 vol.1, 2001. 25

[van Dam 2003] Tijs van Dam and Koen Langendoen. *An Adaptive Energy-efficient MAC Protocol for Wireless Sensor Networks.* In Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys '03, pages 171–180, New York, NY, USA, 2003. ACM. 17

[van der Schaar 2005] Mihaela van der Schaar and Sai Shankar Nandagopalan. *Cross-layer wireless multimedia transmission: challenges, principles, and new paradigms.* IEEE Wireless Commun., vol. 12, no. 4, pages 50–58, 2005. 12

[Čagalj 2002] Mario Čagalj, Jean-Pierre Hubaux and Christian Enz. *Minimum-energy Broadcast in All-wireless Networks: NP-completeness and Distribution Issues.* In Proceedings of the 8th Annual International Conference on Mobile Computing and Networking, MobiCom '02, pages 172–182, New York, NY, USA, 2002. ACM. 15

[Wang 2003] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless and Christopher Gill. *Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks.* In Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys '03, pages 28–39, New York, NY, USA, 2003. ACM. 16

[Wang 2009] Xiaodong Wang, Xiaorui Wang, Xing Fu, Guoliang Xing and Nitish Jha. *Flow-Based Real-Time Communication in Multi-Channel Wireless Sensor Networks.* In Proceedings of the 6th European Conference on Wireless Sensor Networks, EWSN '09, pages 33–52, Berlin, Heidelberg, 2009. Springer-Verlag. 33, 41

[Willett 2004] R. Willett, A. Martin and R. Nowak. *Backcasting: adaptive sampling for sensor networks.* In Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on, pages 124–133, 2004. 12

[Winter] T. Winter, P. Thubert, A. Brandt, T. Clausen, J. hui hui hui hui, R. Kelsey, P. Levis, K. Pister, R. Struik and JP. Vasseur. *Internet draft.* 16

[Wu 1999] Jie Wu and Hailan Li. *On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks.* In Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, DIALM '99, pages 7–14, New York, NY, USA, 1999. ACM. 12

[Wu 2008] Yafeng Wu, J.A. Stankovic, Tian He and Shan Lin. *Realistic and Efficient Multi-Channel Communications in Wireless Sensor Networks.* In INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, pages –, 2008. 26, 33, 36, 39, 41, 67

[Wu 2009] Yafeng Wu, Matthew Keally, Gang Zhou and Weizhen Mao. *Traffic-Aware Channel Assignment in Wireless Sensor Networks*. In Proceedings of the 4th International Conference on Wireless Algorithms, Systems, and Applications, WASA '09, pages 479–488, Berlin, Heidelberg, 2009. Springer-Verlag. 34, 35, 36, 38, 41

[Xing 2007] Kai Xing, Xiuzhen Cheng, Liran Ma and Qilian Liang. *Superimposed Code Based Channel Assignment in Multi-radio Multi-channel Wireless Mesh Networks*. In Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking, MobiCom '07, pages 15–26, New York, NY, USA, 2007. ACM. 25

[Yackovich 2011] John Yackovich, Daniel Mosse, Anthony Rowe and Raj Rajkumar. *Making WSN TDMA Practical: Stealing Slots Up and Down the Tree*. 2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, vol. 1, pages 41–50, 2011. 109

[Ye 2004] Wei Ye, J. Heidemann and D. Estrin. *Medium access control with coordinated adaptive sleeping for wireless sensor networks*. Networking, IEEE/ACM Transactions on, vol. 12, no. 3, pages 493–506, 2004. 17

[Yen 2009] H. H Yen and C. L Lin. *Integrated channel assignment and data aggregation routing problem in wireless sensor networks*. Communications, IET, vol. 3, no. 5, pages 784–793, 2009. 38

[Yu 2010a] Qing Yu, Jiming Chen, Yanfei Fan, Xuemin Shen and Youxian Sun. *Multi-Channel Assignment in Wireless Sensor Networks: A Game Theoretic Approach*. In INFOCOM, 2010 Proceedings IEEE, pages 1–9, 2010. 38

[Yu 2010b] Qing Yu, Jiming Chen, Youxian Sun, Yanfei Fan and Xuemin Shen. *Regret Matching Based Channel Assignment for Wireless Sensor Networks*. In Communications (ICC), 2010 IEEE International Conference on, pages 1–5, 2010. 34, 35, 38, 41

[Zeng 2007] Kai Zeng, Wenjing Lou, Jie Yang and D.R. Brown. *On Geographic Collaborative Forwarding in Wireless Ad Hoc and Sensor Networks*. In Wireless Algorithms, Systems and Applications, 2007. WASA 2007. International Conference on, pages 11–18, 2007. 15

[Zhang 2004a] Wensheng Zhang and Guohong Cao. *DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks*. IEEE TRANSACTIONS ON WIRELESS COMMUNICATION, vol. 3, no. 5, pages 1689–1701, 2004. 12

[Zhang 2004b] Wensheng Zhang and Guohong Cao. *Optimizing Tree Reconfiguration for Mobile Target Tracking in Sensor Networks*. In INFOCOM, 2004. 12

[Zhang 2007] J. Zhang, C. Jeong, G. Lee and H. Kim. *Cluster-Based Multi-Path Routing Algorithm for Multi-hop Wireless Network*. Future Generation Communication and Networking, vol. 1, pages 67–75, 2007. 16

[Zhang 2009] Haibo Zhang, Pablo Soldati and Mikael Johansson. *Optimal link scheduling and channel assignment for convergecast in linear wirelessHART networks.* In Proceedings of the 7th international conference on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOPT'09, pages 82–89, Piscataway, NJ, USA, 2009. IEEE Press. 51, 66

[Zhang 2011] Xiaoling Zhang, Haibin Yu, Wei Liang and Meng Zheng. *Joint routing, scheduling, and power control for multichannel wireless sensor networks with physical interference.* Journal of Control Theory and Applications, vol. 9, no. 1, pages 93–105, 2011. 87

[Zhao 2011] Z. Zhao, G.-H. Yang, Q. Liu, V.O.K. Li and L. Cui. *Implementation and application of a multi-radio wireless sensor networks testbed.* Wireless Sensor Systems, IET, vol. 1, no. 4, pages 191–199, 2011. 28

[Zhou 2006a] Gang Zhou, Tian He, Sudha Krishnamurthy and John A. Stankovic. *Models and Solutions for Radio Irregularity in Wireless Sensor Networks.* ACM Trans. Sen. Netw., vol. 2, no. 2, pages 221–262, May 2006. 29

[Zhou 2006b] Gang Zhou, Chengdu Huang, Ting Yan, Tian He, John A. Stankovic and Tarek F. Abdelzaher. *MMSN: Multi-frequency media access control for wireless sensor networks.* In In IEEE INFOCOM, page 7, 2006. 34, 37, 38, 41, 42

[Zhou 2007] H. Zhou, Ch. Yeh and H.T. Mouftah. *Reliable Low-overhead MAC Protocol for Multi-channel Wireless Mesh Networks.* In IEEE Global Telecommunications Conference (GLOBECOM), 2007. 26

Wireless Sensor Networks in Industrial Environment:
Energy Efficiency, Delay and Scalability

Abstract: Applying WSNs in industrial environment requires fast and reliable data gathering (or data convergecast). If packets are forwarded individually to the sink, it is called raw data convergecast. In this thesis, we resort to the multichannel paradigm to enhance the data gathering delay, the robustness against interferences and throughput. Since some applications require deterministic and bounded convergecast delays, we target conflict-free joint time slot and channel assignment solutions that minimize the schedule length. Such solutions allow nodes to sleep in any slot where they are not involved in transmissions. Hence, they save their limited energy budget.

After a comprehensive study of multichannel assignment protocols, we extend existing results to take into account a sink equipped with multiple radio interfaces and heterogeneous traffic demands. Indeed, we compute the theoretical bounds, that is the minimum number of time slots needed to complete convergecast, in various topologies with different traffic demands. These bounds are provided for different acknowledgment policies. For each of them, we provide a graph-based interference model. We also give optimal schedules that achieve these optimal bounds. We formalize the problem of multichannel slot assignment using integer linear programming and solve with GLPK tool for small configurations. We propose MODESA, a centralized joint time slot and channel assignment algorithm. We prove the optimality of MODESA in lines, multilines and balanced trees topologies. By simulations, we show that MODESA outperforms TMCP, a well known subtree-based scheduling. We improve MODESA with different channel allocation strategies depending on the channel selection criteria (channels load balancing or preference of channels with the best qualities). Moreover, we show that resorting to multipath routing minimizes the convergecast delay. This work is extended in MUSIKA to take into account multi-sinks WSNs and traffic differentiation: the problem is formalized using integer linear programming and solved with GLPK. Simulations results show that the schedule length is minimized and the buffer size is reduced.

We then address the adaptivity and scalability challenges. The slot assignment should be more flexible and able to adapt to application and environment variability (e.g., alarms, temporary additional demands). Theoretical bounds on the number of additional slots introduced to cope with traffic changes, are given. AMSA, an incremental solution, is proposed. Its performances are evaluated in two cases: retransmissions or temporary changes in application needs. To tackle the scalability issue, we propose, WAVE, a simple joint time slot and channel assignment algorithm that operates in centralized or distributed mode. We prove the equivalence of the schedules provided by the two modes.

Keywords: multichannel, wireless sensor networks, data gathering, convergecast, time slot assignment, channel allocation, conflict-free schedule, multi-interfaces, multi-sink