Aasem Ahmad

Optimized TDMA Scheduling Algorithms for Cluster-Tree WSNs

May 2019

CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Electrical Engineering
Department of Control Engineering

Optimized TDMA Scheduling Algorithms for Cluster-Tree WSNs

by

Aasem Ahmad

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

Doctoral Thesis

Supervisor: Prof. Dr. Ing. Zdeněk Hanzálek
Ph.D. programme: Electrical Engineering and Information Technology
Branch of study: Control Engineering and Robotics
Submission date: May 2019

Aasem Ahmad: Optimized TDMA Scheduling Algorithms for Cluster-Tree WSNs, Ph.D. Thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Control Engineering, May 2019, Prague.

Acknowledgement

I express my sincere gratitude to Prof. Dr. Ing. Zdeněk Hanzálek for his invaluable advices, assistance and continuous support leading towards this thesis. I also record my deep sense of gratitude to Doc. Ing. Přemysl Šůcha, Ph.D. for his continuous encouragements and support. I am also grateful to the team members for creating a pleasant, friendly and excellent working environment. Also, I wish to express my appreciation for the Department of Control Engineering and the Czech Technical University in Prague for having the opportunity to study here. Last, but not least, acknowledgments belong to my father, my mother soul, and my brothers for their endless love and support. Without them, none of this would have been possible.

Finally, I acknowledge all the grants which funded the research presented in this dissertation. Namely, the work was supported by the Grant Agency of the Czech Republic under the projects GACR P103/12/1994 and FOREST GACR P103-16-23509S.

Aasem Ahmad
Prague, May 2019

Declaration

This doctoral thesis is submitted in partial fulfillment of the requirements for the degree of doctor (Ph.D.). The work submitted in this dissertation is the result of my own investigation, except where otherwise stated. I declare that I worked out this thesis independently and I quoted all used sources of information in accord with Methodical instructions about ethical principles for writing academic thesis. Moreover I declare that it has not already been accepted for any degree and is also not being concurrently submitted for any other degree.

Aasem Ahmad
Prague, May 2019

Abstract

Wireless Sensor Networks (WSNs) have naturally emerged driven by the recent advances in wireless communications, micro-electro-mechanical systems, and highly integrated electronics. Moreover, the tendency for the integration of computations with physical processes is pushing towards new paradigms such as the Internet of Things (IoT) and Industry 4.0. In that direction, WSNs are becoming more and more valuable enabling infrastructures for a vast range of applications in the domain of modern networked embedded systems such as industrial monitoring and control applications, smart cities, home automation, etc. Such applications tend to connect a tremendous number of small and smart sensor nodes to monitor and control everything, everywhere, even in hard to reach environments. Due to the scarce resources of the sensor nodes (e.g., memory size, processor power, and battery capacity) and due to the specific requirements for the target application, it is very important to develop WSNs having in mind a particular set of Quality of Service (QoS) properties such as collision avoidance, energy efficiency, timeliness, and network reliability. Also, other requirements such as on-the-fly deployment and configuration are essential.

The IEEE 802.15.4/ZigBee standards are leading technologies for low-cost, low-power and low-rate WSNs. Besides, the beacon-enabled IEEE 802.15.4/ZigBee Cluster-Tree WSN topology is one of the infrastructure-based WSNs technology that supports predictable performance and energy efficient behavior that are suited for time-sensitive applications using battery-powered nodes. However, the current state-of-the-art reveals a strong immatureness and an evident lack of solutions concerning the above mentioned stringent required QoS properties. For example, the Cluster-Tree topology, in contrast with the star and mesh topologies, expresses several challenging and open research issues such as a precise cluster schedule that avoids intercluster collisions (messages are transmitted from nodes at different intervals). Furthermore, it is significantly harder and more challenging to obtain the cluster schedule that addresses all the above mentioned QoS properties. In that direction, this thesis contributes to the support of the technology through the design of centralized and distributed, exact and heuristic Time Division Multiple Access (TDMA) cluster scheduling algorithms while considering a realistic system model that relies as much as possible upon the real application scenarios. More specifically, this thesis considers a beacon-enabled Cluster-Tree topology with single-collision domain and multiple-collision domains. The traffic within the network is assumed to be organized into a set of multi-hops data flows, each given by a set of parameters such as source nodes, sink node, and end-to-end deadline. To support control applications where the data goes in both directions simultaneously (i.e., sensed data and control data from and to the field nodes), we deal with data flows that traverse the network simultaneously in opposite directions. The proposed centralized algorithms support collision avoidance, energy efficiency, timeliness, and network reliability. Furthermore, they enable the system designers to efficiently configure all the required parameters of the IEEE 802.15.4/ZigBee beacon-enabled Cluster-Tree WSNs. On the other hand, the proposed distributed algorithms aim to further support the on-the-fly deployment and configuration. Therefore, the distributed algorithms enable each cluster within the network to configure by itself all the required parameters concerning the addressed set of QoS properties. The heuristic algorithms, compared to the more optimal exact algorithms, are proven, by the experimental results, to be efficient in both computation time (large size instances with thousands of nodes are solved in a short time) and solution quality (evaluated over small size instances while comparing it with the optimal solution obtained by the exact algorithms).

Keywords: Cluster-Tree topology; Wireless Sensor Network; IEEE 802.15.4; ZigBee; time bounded data flows; Quality of Service; Time Division Multiple Access, simulation; centralized algorithms; distributed algorithms.

Abstrakt

Bezdrátové senzorové sítě (WSN) se přirozeně objevily díky nedávnému pokroku v oblasti bezdrátové komunikace, mikro-elektro-mechanických systémů a vysoce integrované elektroniky. Navíc tendence k integraci výpočtů s fyzickými procesy směřuje k novým paradigmatům, jako je Internet věcí (IoT) a Průmysl 4.0. V tomto ohledu se WSN stávají stále cennější infrastrukturou pro širokou škálu aplikací v oblasti moderních distribuovaných vestavěných systémů, jako jsou průmyslové monitorovací a řídicí aplikace, inteligentní města, domácí automatizace atd. Tyto aplikace mají tendenci spojovat obrovské počet malých a inteligentních senzorových uzlů pro sledování a řízení všeho, všude i v těžko přístupných prostředích. Vzhledem k omezeným zdrojům senzorových uzlů (např. velikost paměti, výkon procesoru a kapacita baterie) a vzhledem ke specifickým požadavkům na cílovou aplikaci je velmi důležité vyvinout WSNs s ohledem na konkrétní sadu kvality služeb (QoS), jako je zabránění kolizím, energetická efektivita, včasnost doručení zpráv a spolehlivost sítě. Nezbytné jsou také další požadavky, jako je nasazení a konfigurace za provozu zařízení.

Standardy IEEE 802.15.4/ZigBee jsou špičkovými technologiemi pro nízkonákladová zařízení WSN s nízkou spotřebou a malou přenosovou rychlostí. Kromě toho, protokol IEEE 802.15.4/ZigBee Cluster-Tree je jednou z WSN technologií založenou na infrastruktuře, která podporuje předvídatelný výkon a energeticky efektivní chování, které je vhodné pro časově náročné aplikace využívající uzly napájené bateriemi. Současný stav techniky však ukazuje nezralost a zjevný nedostatek řešení týkajících se výše uvedených náročných požadavků na kvalitu služeb. Například Cluster-Tree topologie, na rozdíl od hvězdy a volné topologie, klade několik náročných a otevřených výzkumných otázek, jako je přesný rozvrh, který předchází kolizím mezi klastry (zprávy jsou přenášené z uzlů v různých intervalech). Kromě toho je podstatně těžší a náročnější získat rozvrh klastrů, který splňuje všechny výše uvedené požadavky na kvalitu služeb. V tomto směru tato disertační práce přispívá k podpoře technologie prostřednictvím návrhu centralizovaných a distribuovaných, exaktních a heuristických algoritmů pro časem řízený přístup k přenosovému médiu (TDMA) při volbě realistického modelu systému, který se v maximální možné míře opírá o skutečné aplikace. Konkrétněji se tato práce zabývá Cluster-Tree topologií s jednou kolizní doménou a s více kolizními doménami. Předpokládá se, že provoz v rámci sítě je organizován do datových toků, z nichž každý je dán sadou parametrů, jako jsou zdrojové uzly, cílové uzly, termín doručení a požadovaná perioda.

Pro podporu řídících aplikací, kde data procházejí současně oběma směry (tj. měřené veličiny a řídicí veličiny z a do uzlů v procesu), se zabýváme toky dat, které procházejí sítí současně v opačných směrech. Navrhované centralizované algoritmy podporují předcházení kolizím, energetickou efektivitu, včasnost doručení zpráv a spolehlivost sítě. Dále umožňují systémovým návrhářům efektivně konfigurovat všechny požadované parametry IEEE 802.15.4 / ZigBee Cluster-Tree WSN. Na druhé straně, navrhované distribuované algoritmy jsou zaměřeny na zavádění a konfiguraci za provozu. Distribuované algoritmy proto umožňují, aby každý klastr v síti sám konfiguroval všechny požadované parametry týkající se požadovaných kvalit služeb. Heuristické algoritmy, ve srovnání s více optimálními centralizovanými algoritmy, jsou experimentálními výsledky prokázány jako efektivní s ohledem jak na výpočetní čas (velké instance s tisíci uzly jsou vyřešeny v krátkém čase) tak s ohledem na kvalitu řešení (na malých instancích lze výsledek porovnat s optimálním řešením získaným exaktními algoritmy).

Klíčová slova: Cluster-Tree topoloie; bezdrátové senzorové sítě; IEEE 802.15.4; ZigBee; časově kritické datové toky; kvalita služeb; časem řízený přístup k přenosovému médiu, simulace; centralizovaný algoritmus; distribuovaný algoritmus.

Goals and Objectives

The thesis is aimed to further develop the support for the wireless infrastructure of Internet of Things (IoT) and Industry 4.0 paradigms. Its main goals were determined as follows:

1. Study the existing literature related to the design of TDMA scheduling algorithms for WSNs.
2. Design and implement fast, exact and heuristic, centralized and distributed TDMA scheduling algorithms for static single-collision domain and multiple-collision domains Cluster-Tree WSN with a predefined set of time-bounded data flows, assuming bounded communication errors.
3. The proposed algorithms are required to support a set of Quality of Service (QoS) properties such as collision avoidance, energy efficiency, timeliness, network scalability and reliability, and on-the-fly deployment and configuration.
4. Verify the proposed algorithms on benchmark instances and compare them with the existing works.
5. Implement simulation scenarios using Opnet Modeler 17.5 to further verify the proposed solutions.

Contents

List of Acronyms 1

1 Introduction 3

1.1 Approach and Technology . . . 4
1.2 Outline and Contributions . . . 6
1.3 Structure of the Thesis . . . 9

2 Overview of IEEE 802.15.4 and ZigBee 11

2.1 IEEE 802.15.4/ZigBee Network Devices . . . 11
2.2 Device Roles . . . 12
2.3 IEEE 802.15.4/ZigBee Communication Basics . . . 13
    2.3.1 Beacon-Enabled Operation and Superframe Structure 13
2.4 IEEE 802.15.4/ZigBee Networking topologies . . . 14

3 Optimized TDMA Scheduling Algorithm for Single-Collision Domain ZigBee Cluster-Tree Topology 19

3.1 Related Work . . . 21
3.2 Chapter Outline and Contribution . . . 22
3.3 System Model . . . 23
    3.3.1 Cluster-Tree Topology Model . . . 23
    3.3.2 Data Flow Model . . . 23
    3.3.3 Cluster Life Cycle . . . 25
    3.3.4 Cyclic Nature of the Cluster Schedule . . . 25
3.4 TDMA^{scd} Scheduling Algorithm . . . 29
    3.4.1 Duration of the Cluster's Active Portion . . . 30
    3.4.2 Modeling the Deadline Constraints of the Data Flows 32
    3.4.3 Cluster Partial Ordering Formulated as a Shortest Path Problem . . . 35
    3.4.4 Topological Ordering of POCA Graph . . . 36
3.5 Experimental Results . . . 37
3.6 Simulation Study . . . 37
    3.6.1 Simulation Scenario . . . 38
    3.6.2 Transmission Reliability . . . 39
    3.6.3 Energy Consumption . . . 39
    3.6.4 Timeliness of the Data Flows . . . 40
3.7 Conclusion . . . 41

4 Optimized TDMA Scheduling Algorithms for Multiple-Collision Domains ZigBee Cluster-Tree Topology 43
4.1 Related Work 44
4.2 Chapter Contributions and Outline 45
4.3 System Model 46
4.3.1 Multiple Collision Domains Model 47
4.4 TDMA^{mcd} Cluster Scheduling Algorithm 47
4.4.1 E_TDMA^{mcd} Scheduling Algorithm 49
4.4.2 H_TDMA^{mcd} Scheduling Algorithm 51
4.5 Experimental Results 54
4.5.1 Benchmarks Settings 54
4.5.2 Computation Time of the Algorithms 54
4.5.3 Success Rate of the Algorithms 56
4.5.4 Solution Quality of the H_TDMA^{mcd} Algorithm 58
4.6 Simulation Study 59
4.6.1 Simulation Scenarios 59
4.6.2 Network Reliability 59
4.6.3 Energy Consumption 60
4.6.4 Timeliness of the Data Transmission 61
4.7 Conclusion 61

5 Optimized Distributed TDMA Scheduling Algorithms for ZigBee-Like Cluster-Tree Topology 63
5.1 Related Work 64
5.2 Chapter Contributions and Outline 66
5.3 System Model 67
5.3.1 Cluster-Tree Topology 67
5.3.2 Data Flow Model 68
5.3.3 Cluster Life cycle 68
5.3.4 Multiple-Collision Domains 69
5.3.5 Cyclic Nature of the Cluster Schedule 70
5.4 Distributed TDMA Scheduling Algorithm 71
5.5 The Calculation of P_{max} and the Number of Clusters m . . . 72
5.6 Determining the Precedence Decisions 75
5.6.1 Construction of the Inequality Graph 76
5.6.2 The Calculation of D_i 78
5.7 Calculation of the Active Portion Duration and the Number of Tx and Rx Slots: 85
5.8 Cluster Offset Within the Schedule Period 86
5.8.1 Single-Collision Domain Case Study 87
5.8.2 Multiple-Collision Domains Case Study 88
5.9 Parent-child time-slot Allocation 94

5.10 Experimental Results 95
5.10.1 Benchmarks Settings 95
5.10.2 Success Rate of the Algorithms 96
5.10.3 Computation Time of the Algorithms 98
5.10.4 Number of Transmitted Packets 101
5.10.5 Energy Consumption 103
5.10.6 Network Reliability 105
5.11 Conclusion 105

6 Conclusion **107**

Bibliography **116**

A Curriculum Vitae **117**

B List of Author's Publications **119**

List of Figures

2.1 ZigBee Wireless Networking Protocol Layers. 12
2.2 Device Roles in the IEEE 802.15.4 and ZigBee Standards. . . 12
2.3 Superframe structure. 14
2.4 IEEE 802.15.4/ZigBee network topologies. 16

3.1 Cluster-Tree topology with 4 time bounded data flows. 20
3.2 Cluster-graph that shows the source and sink clusters of each data flow in Fig. 3.1. 24
3.3 Superframe structure. 25
3.4 Chain network with two GTSs allocations scenarios. 27
3.5 Feasible cyclic schedule for the example shown in Fig. 3.1. . . 29
3.6 The Inter-Frame Spacing. 32
3.7 Precedence decisions determination. 33
3.8 The topological and data flows constraints. 34
3.9 The constraints for the example presented in Fig. 3.1. 35
3.10 The inequality graph representation for the constraint model shown in Fig. 3.9. 36
3.11 The simulation scenario for the example presented in Fig. 3.1. 39
3.12 The impact of the number of re-transmissions on the network reliability and the energy consumption for given length of the schedule period. 40
3.13 The impact of the number of re-transmissions and the length of the schedule period on the end-to-end delay of the data flows. 41

4.1 Feasible cyclic schedule for the example shown in Fig. 3.1 with the spatial reuse of the transmission medium. 48
4.2 The constraint model. 50
4.3 The topological and data flows constraints for the example presented in Fig. 3.1. 52
4.4 Precedence decisions determination. 52
4.5 The Task graph that is based on POCA graph in Fig. 4.4b. . 52
4.6 The evaluation of TDCS, E_TDMA^{mcd} and H_TDMA^{mcd} algorithms. 57
4.7 The impact of the number of re-transmissions on the network reliability and the energy consumption for given length of the schedule period. 60

4.8 The impact of the number of re-transmissions and the length of the schedule period on the end-to-end delay of the data flows. 61

5.1 An example of Cluster-Tree WSNs with four time-constrained data flows. 64
5.2 Cluster-graph that shows the source and sink clusters of each data flow in Fig. 5.1. 69
5.3 Cluster life cylce. 70
5.4 Number of clusters and period calculation. 74
5.5 The constraint model for the example in Fig. 5.1. 75
5.6 Determination of the precedence decisions. 75
5.7 The distributed construction of the inequality graph. (a): a Cluster-Tree topology with 6 nodes and two data flows (b): transmission of FLOW-INFO packet from node 4 to node 6, (c): transmission of FLOW-ACK from node 6 to node 4, (d): transmission of FLOW-INFO from node 6 to node 1, (e): transmission of FLOW-ACK from node 1 to node 6, (f): the order of the nodes in sending the READY packets, (g): the user defined data flows parameters for the example in Fig. 5.7a. (h): FLOW-INFO packet format, (i): FLOW-ACK packet format. 80
5.8 Distributed calculation of D_i for the example shown in Fig. 5.7, (a)...(f) the first iteration of packet transmissions, (g)...(i) the second iteration of packet transmissions, (m) transmission of STOP packet, (n) the SHDV packet format, (o) the MHDV packet format, (p) the READY packet format, (q) the STOP packet format. 84
5.9 MAC frame. 86
5.10 Distributed topological ordering during the calculation of the cluster offset stage of DTDMA$^{\mathrm{scd}}$. 88
5.11 The cluster-head node finite state transition diagram during the calculation of the cluster offset stage of DTDMA$^{\mathrm{mcd}}$ algorithm. (1): $\mathit{Pred} = \emptyset$, (2): $\mathit{Pred} \neq \emptyset$, (3): $\mathit{Pred} = \emptyset$, (4): All PROBE-NOTIFY packets are sent, (5): all feedbacks are received and the current node has the maximum ID among the nodes in *Rel* set, (6): all READY packets have been received from all child nodes and the node has sent READY packet to its parent node. 90
5.12 The changes on the cluster-head nodes states while applying the calculation of the cluster offset stage of DTDMA$^{\mathrm{mcd}}$ for the POCA graph shown in Fig. 5.6b 91

5.13 The *parent-child* time-slots allocation stage for the example shown in Fig. 5.10 93
5.14 A cluster schedule of the example shown in Fig. 5.1. 94
5.15 The evaluation of DTDMA$^{\mathrm{s}}$, DTDMA$^{\mathrm{scd}}$, DTDMA$^{\mathrm{mcd}}$ and TDCS algorithms when $loss\_rate = 30\%$ and $avg(|M(C_u)|) \leq 25$. 99
5.16 The performance evaluation of DTDMA$^{\mathrm{mcd}}$ as a function to the *loss_rate* parameter. 100
5.17 The performance evaluation of DTDMA$^{\mathrm{mcd}}$ as a function to the competitor set. 101
5.18 The energy consumption of the network within 60 min based on the cluster-schedule obtained by both DTDMA$^{\mathrm{mcd}}$ and TWBS as a function to the value of PO and the number of clusters. 104
5.19 The impact of the loss rate on the network reliability 104

List of Tables

2.1 Star vs. mesh vs. Cluster-Tree topologies. 15

3.1 The user-defined parameters of the data flows from Fig. 3.1 (ptu = processing time unit). 20
3.2 Three scenarios illustrating the impact of the precedence decisions on the θ_{f_q} for each data flow f_q in Fig. 3.1. 33
3.3 Time complexity of the TDMA$^{\text{scd}}$ algorithm. 38

4.1 Time computation and solution quality. 55

5.1 User-defined data flows parameters. 64
5.2 The impact of the precedence decisions on θ_{f_q}. 71
5.3 The initial value of PO_{max} and m for the example in Fig. 5.1. 74
5.4 The evaluation of DTDMA$^{\text{scd}}$ and DTDMA$^{\text{mcd}}$ algorithms when $loss\_rate = 30\%$ and $avg(|C_m|) \leq 25$. 98

List of Algorithms

1 The TDMA^{scd} algorithm. 30
2 The calculation of the Superframe duration. 32
3 The determination of the POCA graph edges. 36

4 The TDMA^{mcd} algorithm. 49
5 The H_TDMA^{mcd} scheduling algorithm. 53

6 The distributed calculation of the number of clusters and the upper bound of the schedule period. 73
7 The construction of the inequality graph 79
8 The distributed calculation of D_i. 82
9 The distributed determination of the successor and the predecessor clusters. 83
10 The distributed calculation of the active portion and time-slots durations. 86
11 The centralized algorithm utilized over the cluster offset calculation. 89
12 The centralized cluster offset adjustment algorithm. 90
13 The distributed algorithm utilized over the cluster offset calculation. 92
14 The distributed cluster offset adjustment algorithm. 93

List of Acronyms

AODV	Ad hoc On-Demand Distance Vector	4
BI	Beacon Interval	14
BO	Beacon Order	14
CAP	Contention Access Period	13
CCA	Clear Channel Assessment	13
CFP	Contention Free Period	13
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance	13
DAC	Directed Acyclic Graph	36
DV	Distance Vector	78
ETO	End Topological Ordering	87
FFD	Full-Function Device	11
GTS	Guaranteed Time Slot	13
IFS	Inter-frame Spacing	30
ILP	Integer Linear Programming	7
IoT	Internet of Things	3
LIFS	Long Inter-Frame Spacing	31
MAC	Medium Access Control	4
MHDV	Multiple Hops Distance Value	81
P-NC	Period-Number of Clusters	74
PAN	Personal Area Network	12
POCA	Partial Order of Cluster Activations	22
QoS	Quality of Service	3
RFD	Reduced-Function Device	11
RP-NC	Required Period-Number of Clusters	73
SD	Superframe Duration	14
SHDV	Single Hop Distance Value	81
SIFS	Short Inter-Frame Spacing	31
SO	Superframe Order	14
STO	Start Topological Ordering	87
TDCS	Time Division Cluster Schedule	8

TDMA	Time Division Multiple Access	4
TSA	Time-Slot Allocation	94
TWBS	Two Way Beacon Scheduling	66
WSA	Window Scheduling Algorithm	21
WSNs	Wireless Sensor Networks	3
ZC	ZigBee Coordinator	12
ZED	ZigBee End Device	12
ZR	ZigBee Router	12

Chapter 1

Introduction

Recent advances in wireless communications, micro-electro-mechanical systems, and highly integrated electronics have driven the emergence of Wireless Sensor Networks (WSNs). Moreover, the tendency for the integration of computations with physical processes is pushing towards new paradigms such as the Internet of Things (IoT) and Industry 4.0. In that direction, WSNs are becoming more and more valuable enabling infrastructure for a vast range of applications in the domain of modern networked embedded systems such as industrial monitoring and control applications, smart cities, home automation, etc [46, 23]. The use of WSNs for such systems, in contrast to wired solutions, provides a solution that increases efficiency and reduces cost since the installation and maintenance of cables are usually much more expensive than the cost of the sensors themselves.

The modern networked embedded systems tend to connect a tremendous number of smart and small sensor nodes that are typically deployed in large size area in order to provide sensing and actuating actions even in hazardous environments and hard to reach regions [48, 2]. However, due to the scarce resources of the sensor nodes (e.g., memory size, processor power, and battery capacity) and due to the specific requirements for the target application, a rethinking of the usual computing and networking concepts is crucial. In particular, novel optimized concepts and solutions are needed in order to guarantee and improve a particular set of Quality of Service (QoS) properties [43, 36, 16]. Specifically, the collision avoidance among the sensor nodes is a critical issue to be resolved in order to coordinate the access to the shared wireless transmission medium (i.e., the frequency channel). Moreover, since wireless nodes are usually battery powered, then energy efficiency is a problem of paramount importance in order to prolong the lifetime of the network. Also, for the applications relying on the transmission of time-sensitive messages (e.g., real-time tracking systems), the end-to-end delay of those messages must be bounded. Therefore, the timeliness QoS property for the traffic within the network is important to be supported [55, 14]. Moreover, due to different kinds of disturbances, the transmission over the frequency channel might be lost or corrupted. Hence, the reliability of the data transmissions must be considered to ensure, with a high probability as specified by the target application, that each message reaches the specified recipient logically correct and on time [19]. Also, on-the-fly deployment and configuration QoS is essential in order to enable each sensor node to configure by itself all the required parameters when the nodes are deployed in hard to reach regions.

In this thesis, we aim to further support the current technology by providing novel optimized techniques/algorithms that lead to the improvement of the previously mentioned QoS properties. In that direction, we clearly understand that developing such optimized techniques/algorithms for WSNs is a prominent challenging problem. For instance, it is unrealistic to support hard real-time communications in a WSN due to communication errors resulting from the unreliable and time-varying characteristics of wireless channels [25]. Furthermore, the QoS properties are correlated in the sense that the sound solution that improves a particular QoS property might degrade the other QoS properties. For example, to improve network reliability, the re-transmissions and acknowledgment mechanism is usually used. However, such approach drains the battery of the node faster and consequently, the energy efficiency QoS property degrades. Therefore, a fair trade-off solution should be found in order to address both network reliability and energy efficiency. Such a fair solution can be verified using a simulation model prior to the network deployment.

1.1 Approach and Technology

WSNs can be categorized into two types, ad hoc infrastructure-less networks, and infrastructure-based networks. The ad hoc type is characterized by its flexibility and can easily adapt to network changes, but at the cost of unpredictable performance. This is due to the use of contention-based Medium Access Control (MAC) and probabilistic routing protocols such as Ad hoc On-Demand Distance Vector (AODV) [41, 44]. The infrastructure-based type, on the other hand, is less flexible since it employs the pre-deployed and structured topology, but provides better support for the predictable performance guarantees. This is due to the use of synchronization mechanisms and deterministic routing protocols in addition to the support of the contention-free MAC protocols (e.g., Time Division Multiple Access (TDMA)). Hence, since the predictable performance guarantees, as specified by the set of the QoS properties, are the objective of our thesis, we rely on infrastructure-based networks.

One of the infrastructure-based WSNs technology that provides the support for both performance guarantees and energy efficiency is a beacon-enabled IEEE 802.15.4/ZigBee Cluster-Tree WSN topology [37, 7]. The Cluster-Tree topology is a tree based network and the IEEE 802.15.4/ZigBee standards are leading technologies for low-cost, low-power and low-rate WSNs [1, 2]. However, the Cluster-Tree topology expresses many challenging and open research issues in the area of collision avoidance, energy efficiency and real-time communications (e.g., the design of collision-free

TDMA cluster scheduling algorithm that specifies the precise time for the transmitted beacon/message by each node such that the timeliness requirements are met and the lifetime of the network is prolonged).

The design of TDMA cluster scheduling algorithms, while considering Cluster-Tree WSN topology, has been addressed in this thesis in order to provide support to the specified QoS properties (i.e., collision avoidance, energy efficiency, timeliness, network reliability, and on-the-fly deployment and configuration). In particular, the TDMA algorithms slice the time domain into equal sized time-slots so that nodes may be allocated distinctive time-slots [34, 38]. In case of single-collision domain WSNs, each time-slot can be allocated, at most, to one node (i.e., time-slot cannot be shared). However, for multiple-collision domains WSNs, the non-interfering nodes may share the same time-slots (i.e., maximizing the spatial reuse of the available bandwidth while simultaneously eliminating the possibility of collisions) [21]. The number of time-slots assigned to one node is in proportion to the amount of data to be sent by the node. This enables the node to enter power-saving mode until its allocated time-slots in order to save energy [50, 33]. Since TDMA mechanism aims at the elimination of the collision occurrence and seeks to minimize the number of time-slots assigned to each node, the energy consumption of the nodes is also reduced. Furthermore, since the TDMA mechanism provides the ability to perform end-to-end resource reservation of the time-slots, then with the proper ordering of the allocated time-slots to the nodes, the end-to-end delay of the data transmissions can be bounded, and consequently the timeliness QoS can be guaranteed [3, 7, 39].

However, we need to state clearly that the support of both energy efficiency and timeliness is prominently challenging since the energy efficiency and timeliness are correlated in the sense that improving one property might degrade the other one. For example, prolonging the duration at which each node in the network is in power-saving mode maximizes the lifetime of the node battery. However, the longer the duration of the power-saving mode, the less the duration of the duty cycle of the node and consequently, the end-to-end delay of the messages is prolonged. Therefore, a fair trade-off solution must be found and verified with respect to the specified application.

The network reliability is usually provided by the network communication protocols, e.g., at the data link layer, which can detect most of the communication errors and, in some cases, correct some of them. A corrupted or lost message can be detected by simple checksum or acknowledgment mechanisms, respectively, and it can be restored by a re-transmission mechanism, for example. These mechanisms are supported by IEEE 802.15.4 standards. Hence, given the communication error parameter over the frequency channel and the required reliability, the required number of re-transmissions can be calculated so that each message reaches the destination with certain probability as specified by the required reliability. Based on the specified number of re-transmissions, the TDMA mechanism can allocate the required number of time-slots to the each node. Hence, the number of re-transmissions must be bounded even for unknown communication error parameter. Furthermore, since each re-transmission decreases the network throughput and increases the energy consumption and the end-to-end delay of the transmitted data, a fair trade-off is required between reliability, timeliness and energy efficiency QoS properties with respect to the specified application.

The WSNs might be deployed on-the-fly without the need for any pre-existing infrastructure especially in hard to reach environments. In such scenario, the sensor nodes are expected to self organize and configure themselves into the form of a multi-hops network (e.g., Cluster-Tree topology) so they can operate unattended [54, 5]. The realization of such a requirement relies mostly on the distributed methodologies that enable each node within the network to set all required parameters based on its local view of the network [30, 31, 3]. Therefore, in this thesis, we also consider the design of distributed TDMA cluster scheduling mechanisms that enable each node within the network to come up with its allocated time-slots and configure all other parameters, such as the number of re-transmissions and the duration of the power-saving mode, in order to meet all the specified QoS properties.

1.2 Outline and Contributions

The motivation that has driven the work presented in the thesis is the fact that IEEE 802.15.4/ZigBee standards are leading technologies for low-cost, low-power and low-rate WSNs. Besides, the beacon-enabled IEEE 802.15.4/ZigBee Cluster-Tree WSN topology is one of the infrastructure-based WSNs technology that supports predictable performance and energy efficient behavior that are suited for time-sensitive applications using battery-powered nodes. However, the Cluster-Tree topology expresses several challenging and open research issues such as the design of cluster scheduling algorithm that specifies the time-slots allocated to the nodes so that the required QoS properties are met. In that direction, this thesis contributes to the support of the technology through the design of collision-free TDMA cluster scheduling algorithms. The algorithms address the specified QoS properties directly while considering a realistic system model that relies as much as possible upon the real application scenarios. More specifically, this thesis considers beacon-enabled IEEE 802.15.4/ZigBee Cluster-Tree WSN topology. The traffic within the network is assumed to be organized into a set of multi-hops data flows, each given by a set of parameters such as source nodes, sink node, and end-to-end deadline. To support control applications where the data goes in both directions simultaneously (i.e., sensed data and control data from and to the field nodes), we deal with data flows that traverse the network simultaneously in opposite directions. The complexity of the TDMA cluster scheduling problem while considering multiple-collision domains Cluster-Tree topology is $\mathcal{NP}$-hard [22].

Three main parts can be identified in this thesis. The first part, presented in Chapter 3, is related to our work presented in [8] and provides the following contributions:

1. Simplifying the collision-free TDMA cluster scheduling problem by assuming single-collision domain Cluster-Tree topology (i.e. at most, one cluster can be active at any given time) and by expressing the precise end-to-end deadline, given in time units, into the maximum number of crossed periods. Both assumptions lead to polynomial time complexity instead of $\mathcal{NP}$-hard complexity.
2. Proposing and implementing an optimal polynomial Time Division Multiple Access for single-collision domain ($\mathrm{TDMA}^{\mathrm{scd}}$) cluster scheduling algorithm that is based on graph theory algorithms such as shortest path and topological ordering algorithms. The algorithm ensures collision avoidance, energy efficiency, timeliness, and reliability QoS properties.
3. Solving large size instances in a short time.
4. Simulation scenarios are accomplished in Opnet Modeler 17.5 to demonstrate the impact of the number of re-transmissions on the network reliability and timeliness of the data flows. Also, the energy consumption as a function to the number of re-transmissions and the duty cycle of the nodes is demonstrated.

The second part, presented in Chapter 4, is related to our work presented in [7]. Thus, Chapter 4 extends and completes the work presented in Chapter 3 and provides the following contributions:

1. A realistic model with multiple-collision domains Cluster-Tree topology and multi-hops data flows constrained by end-to-end deadlines expressed by the maximum number of crossed periods.
2. Proposing and implementing an optimal Exact Time Division Multiple Access for multiple-collision domains ($\mathrm{E\_TDMA}^{\mathrm{mcd}}$) cluster scheduling algorithm that is based on Integer Linear Programming (ILP). The algorithm addresses collision avoidance, energy efficiency, timeliness and reliability QoS properties for small-size instances.
3. Proposing and implementing Heuristic Time Division Multiple Access for multiple-collision domains $\mathrm{H\_TDMA}^{\mathrm{mcd}}$ cluster scheduling algo-

rithm that is based on a sound formulation of problems and subproblems concerning combinatorial optimization and graph theory. The algorithm addresses collision avoidance, energy efficiency, timeliness and reliability QoS properties for large-size instances with thousands of nodes.
4. The comparison with the Time Division Cluster Schedule (TDCS) algorithm presented in [25] and the evaluation over large-scale benchmarks are demonstrated.
5. Simulation scenarios are accomplished in Opnet Modeler 17.5 in order to demonstrate the correlation among several QoS properties such as reliability, energy efficiency, and timeliness.

The third part, presented in Chapter 5, is related to our work presented in [3, 4] and it aims to further support the on-the-fly deployment and configuration. Therefore, it proposes the following contributions:

1. A realistic model with single-collision domain or multiple-collision domains Cluster-Tree topology and multi-hops data flows constrained by end-to-end deadlines expressed by the maximum number of crossed periods.
2. An exact Distributed Time Division Multiple Access for single-collision domain ($\text{DTDMA}^{\text{scd}}$) cluster scheduling algorithm. $\text{DTDMA}^{\text{scd}}$ algorithm outperforms the distributed algorithm presented in [3] in terms of energy efficiency and computation time.
3. A novel centralized heuristic algorithm that solves the time-slots allocation for the case of multiple-collision domains Cluster-Tree topology. The value of the proposed centralized algorithm, compared to the one proposed in Chapter 4, lies in its ability to be easily distributed without imposing extra calculation overheads.
4. A heuristic Distributed Time Division Multiple Access for multiple-collision domains ($\text{DTDMA}^{\text{mcd}}$) cluster scheduling algorithm which is based on our proposed centralized heuristic algorithm for the time-slots allocation sub-problem.
5. The proposed algorithms are based on the sound formulation of the problems and sub-problems concerning combinatorial optimization and graph theory. Moreover, the algorithms address the collision avoidance, energy efficiency, timeliness, network reliability, and on-the-fly deployment and configuration QoS properties.
6. The evaluation and the comparison of the proposed algorithms with the existing works over large-scale benchmarks, through various simulation scenarios, in order to demonstrate the overhead of the algorithms in term of the elapsed time to construct the schedule, the energy consumption, and network reliability.

1.3 Structure of the Thesis

This thesis is organized as follows. Since the proposed general methodologies are applied to the specific case of IEEE 802.15.4/ZigBee beacon-enabled Cluster-Tree WSNs, Chapter 2 gives an overview to the most significant features of the IEEE 802.15.4 standard and ZigBee specification. Assuming a static single-collision domain Cluster-Tree WSN with a set of time-bounded data flows, Chapter 3 presents a centralized exact TDMA scheduling algorithm that addresses collision avoidance, energy efficiency, timeliness, network reliability and solves large size instances. The objective of Chapter 4 is to extend and complete Chapter 3 by presenting centralized TDMA scheduling algorithms, exact and heuristic, for large scale multiple-collision domains Cluster-Tree WSNs. Chapter 5 proposes distributed TDMA cluster scheduling algorithm for ZigBee-Like Cluster-Tree topology with single-collision domain and multiple-collision domains. The distributed algorithms address, besides the set of the QoSs targeted by the centralized algorithms, the on-the-fly deployment and configuration QoS property. Finally, the conclusions are drawn in Chapter 6.

Complete list of my published/submitted papers is given at the end of the thesis in Section *Author's publications*.

Chapter 2

Overview of IEEE 802.15.4 and ZigBee

THIS chapter is an introduction to the most significant features of the IEEE 802.15.4 standards [2] and ZigBee specifications [9] that are relevant to the thesis. In particular, it focuses on the beacon-enabled Cluster-Tree topology that guarantees predictable QoS properties.

The IEEE 802.15.4/ZigBee wireless technology is the leading global standard for implementing low-cost, low-data-rate, and short-range wireless networks with extended battery life. Sometimes, people may confuse IEEE 802.15.4 with ZigBee. The IEEE 802.15.4 standards specify both the physical layer and MAC sub-layer, while ZigBee specifications provide the network layer and the framework for the application layer so that a full protocol stack is defined [44] as shown in Fig. 2.1. The ZigBee Alliance and the IEEE decided to join forces and ZigBee is the commercial name for the IEEE802.15.4/ZigBee communication technology.

The ZigBee-based wireless devices operate in 868 MHz, 915 MHz, and 2.4 GHz frequency bands. The maximum data rate is 250 Kbits per second. Thus, ZigBee is targeted mainly for battery-powered applications where low data rate, low cost, and long battery life are the main requirements. In many ZigBee applications, the devices have duty cycles of less than 1% to ensure years of battery life [44]. Therefore, the total time the wireless device is engaged in any activity is minimal (i.e., the device spends most of its time in a power-saving mode, also known as sleep mode). As a result, ZigBee-based devices are capable of being operational for several years before their batteries need to be replaced. Furthermore, the standards are quite flexible and can support wide range of applications, e.g., time-sensitive WSN applications, by adequately tuning their parameters.

2.1 IEEE 802.15.4/ZigBee Network Devices

There are two types of devices in an IEEE 802.15.4/ZigBee wireless network: Full-Function Device (FFD) and Reduced-Function Device (RFD). A FFD is capable of communicating with any other device in a network and can accept any role in the network. On the other hand, RFD has limited capabilities and can communicate only with a FFD. RFDs are intended for very simple applications such as turning on or off a switch. The processing power and memory size of RFDs are normally less than those of FFDs [44].

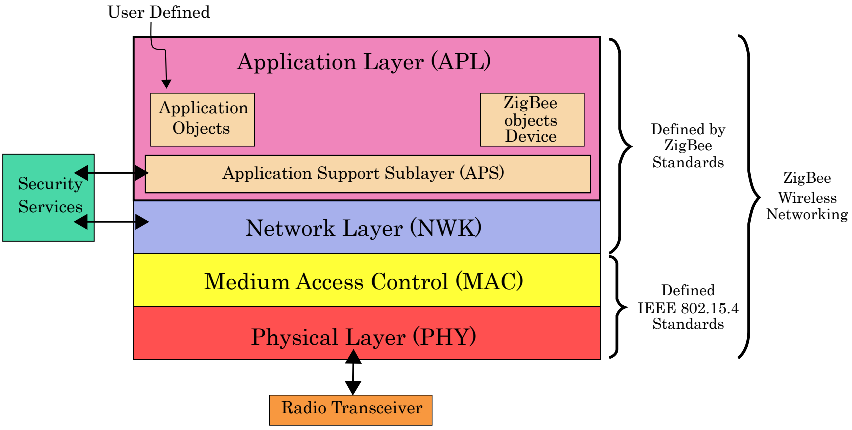


Figure 2.1: ZigBee Wireless Networking Protocol Layers.

2.2 Device Roles

As indicated by the IEEE 802.15.4 standards, a FFD can take three different roles: coordinator, Personal Area Network (PAN) coordinator, and end device. A coordinator is a FFD that participates in multi-hops transmissions (i.e., it is capable of relaying messages). A PAN coordination is a coordinator that is also the principal controller of the PAN. The end device, is simply a device that cannot participate in multi-hops transmissions.

The ZigBee standard uses slightly different terminology (see Fig. 2.2). A ZigBee Coordinator (ZC) is an IEEE 802.15.4 PAN coordinator. A ZigBee Router (ZR) is a device that can act as an IEEE 802.15.4 coordinator. Finally, a ZigBee End Device (ZED) is a device that is neither a coordinator nor a router. A ZED has the least memory size and fewest processing capabilities and features.

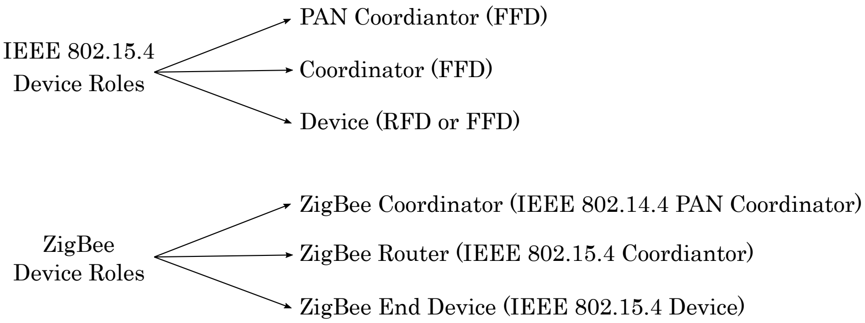

IEEE 802.15.4 Device Roles
PAN Coordiantor (FFD)
Coordinator (FFD)
Device (RFD or FFD)
ZigBee Device Roles
ZigBee Coordinator (IEEE 802.14.4 PAN Coordinator)
ZigBee Router (IEEE 802.15.4 Coordiantor)
ZigBee End Device (IEEE 802.15.4 Device)

Figure 2.2: Device Roles in the IEEE 802.15.4 and ZigBee Standards.

2.3 IEEE 802.15.4/ZigBee Communication Basics

There are two methods for channel access as defined by the MAC layer: contention-based or contention-free. In the contention-based channel access, the Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) mechanism is used by all the devices within the network. The CSMA-CA is a simple mechanism that allows multiple devices to use the same frequency channel as a communication medium. Whenever a device needs to transmit, it first performs a Clear Channel Assessment (CCA) to ensure that the channel is not in use by any other device. Then the device starts transmitting its own signal. If the channel is not clear, the device backs off for a random period of time and tries again. The random back-off and retry are repeated until either the channel becomes clear or the device reaches its user-defined maximum number of retries. Hence, There is no guarantee for any device to use the frequency channel exactly when it needs it.

On the other hand, in the contention-free method, the coordinator may dedicate a specific time-slot, called Guaranteed Time Slot (GTS), to a particular device so that the specified device starts transmitting during that GTS without using the CSMA-CA mechanism. This is a great option for low-latency applications in which the device cannot afford to wait for a random and potentially long period of time until the channel is available as in CSMA-CA.

The allocation of the GTSs requires that the coordinator sends a beacon frame to the specified device in order to ensure clocks synchronization. A network in which each coordinator transmits beacon frame is known as a beacon-enabled network. The beacon-enabled networks can guarantee real-time performance for time-sensitive WSN applications through the GTS allocation. On the other hands, the network at which coordinators do not transmit beacons is known as a non-beacon network. A non-beacon network uses CSMA-CA and cannot have GTSs since the devices are not synchronized. Consequently, non-beacon network cannot support time-sensitive WSN applications.

2.3.1 Beacon-Enabled Operation and Superframe Structure

This thesis considers only the beacon-enabled mode in order to provide support to the time-sensitive WSNs applications. In a beacon-enabled network, each coordinator within the network periodically transmits a beacon frame and uses the superframe structure as shown in Fig. 2.3. The superframe structure is bounded by two beacon frames and can be up to three types of portions: the Contention Access Period (CAP), the Contention Free Period (CFP), and the inactive portion. During CAP, the CSMA-CA

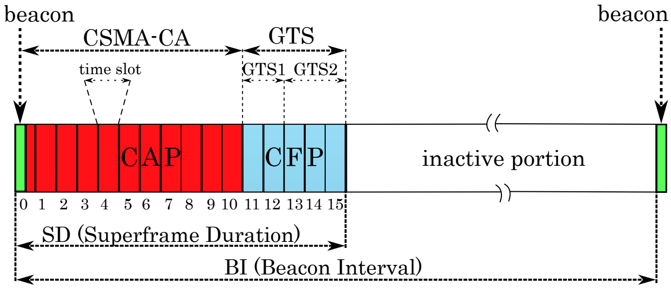


Figure 2.3: Superframe structure.

mechanism is used to gain access to the channel for best-effort data delivery. During CFP, the coordinator may dedicate a specific GTS time-slot(s) to a particular device for real-time transmissions. The combination of CAP and CFP is known as the active portion. The active portion is divided into 16 equal time-slots. The beacon frame always starts at the beginning of the first time-slot. There can be up to 7 GTSs in CFP. Each GTS can occupy one or more time-slots. The inactive portion allows the coordinator to turn off its transceiver circuits to conserve battery energy (i.e., power-saving mode). The inactive portion might be void.

The length of the active and inactive portion, as well as the length of a single time-slot and the usage of GTS slots, are configurable. The duration between two consecutive beacons, known as Beacon Interval (BI), and the length of the active portion of the superframe, known as the Superframe Duration (SD), are defined by two parameters, the Beacon Order (BO) and the Superframe Order (SO) as follows:

$$\begin{aligned} \mathrm{BI} &= \mathit{aBaseSuperframeDuration} \cdot 2^{\mathrm{BO}} \\ \mathrm{SD} &= \mathit{aBaseSuperframeDuration} \cdot 2^{\mathrm{SO}} \end{aligned} \tag{2.1}$$

where $0 \leq \mathrm{SO} \leq \mathrm{BO} \leq 14$ and $\mathit{aBaseSuperframeDuration} = 15.36$ ms (assuming the 2.4 GHz frequency band and 250 kbps of bit rate) and denotes the minimum duration of active portion when $\mathrm{SO} = 0$. Note that the ratio SD/BI is called the duty-cycle.

2.4 IEEE 802.15.4/ZigBee Networking topologies

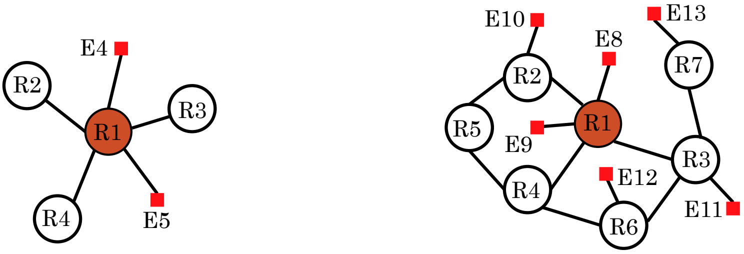
An IEEE 802.15.4/ZigBee network, regardless of its topology, is always created by a PAN coordinator. There is only one PAN coordinator in the entire network which establishes the network. The IEEE 802.15.4/ZigBee standards support three networking topologies: star, mesh and Cluster-Tree. The network toplogies are shown in Fig. 2.4 and the major deferences are summarized in Table. 2.1. The star and the Cluster-Tree networks, in contrast to the mesh topology, can operate on beacon-enabled mode.

Table 2.1: Star vs. mesh vs. Cluster-Tree topologies.

features	**star**	**mesh**	**Cluster-Tree**
scalability	no	yes	yes
energy efficiency	yes	no	yes
network synchronization	yes	no	yes
redundant paths	no	yes	no
deterministic routing	yes	no	yes
contention-free medium access	yes	no	yes

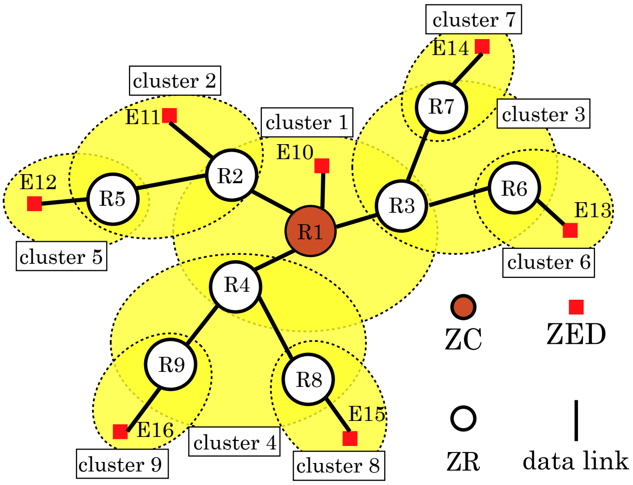
In the star topology, shown in Fig. 2.4a, the ZC (i.e., the PAN coordinator) is activated and starts establishing the network by selecting a unique PAN identifier that is not used by any other nearby network. The other devices join the network by the association with the ZC. The communications are centralized so that every device in the star network can communicate only with the ZC. If a ZED needs to transfer data to another ZED, it sends its data to the ZC, which subsequently forwards the data to the intended recipient. To synchronize the clock among the devices within the network, ZC periodically emits beacon frames. Consequently, ZED may request for the GTS ensuring predictable and contention-free medium access. Furthermore, each ZED can enter a power-saving mode to save its energy whenever it is not engaged in any transmission. The main advantages of the star topology are its simplicity, predictable performance and energy efficient behavior. The disadvantage of this topology is the operation of the network depends on the ZC. In particular, since all transmissions between devices must go through ZC, the ZC may become bottlenecked and single point of failure. Moreover, the battery resource of the ZC can be also rapidly ruined. Therefore, star networks are suitable for simple and small scale applications.

Infrastructure-less mesh topology allows more complex network formations to be implemented. In a mesh topology, shown in Fig. 2.4b, any coordinator in the network can play the role of the PAN coordinator. One way to decide which device will be the PAN coordinator is to pick the first FFD that starts communicating as the PAN coordinator. The ZRs participate in relaying the messages while ZEDs are not capable of relaying the messages. However, ZEDs can be part of the network and communicate only with one particular device (a ZC or a ZR). Thus mesh network differs from the star topology in that the communications are decentralized and each device can communicate directly with any other device if the devices are placed close enough to establish a successful communication link. Moreover, the mesh topology provides good scalability and enhanced



(a) Star topology.

(b) Mesh topology.



(c) Cluster-Tree topology.

Figure 2.4: IEEE 802.15.4/ZigBee network topologies.

network flexibility such as redundant routing paths by the utilization of probabilistic routing protocol (e.g., AODV defined in ZigBee). Therefore, the mesh topology is self-healing, meaning during transmission; if a path fails, the node will find an alternate path to the destination. Consequently, the end-to-end reliability of data transmission is increased and the single point of failure is eliminated. On the other hand, the AODV routing protocol together with the contention-based MAC protocol causes unpredictable end-to-end connectivity between nodes (i.e., unpredictable performance and resource bounds). Moreover, since the routing paths cannot be predicted in advance, the nodes cannot enter power-saving mode which leads to a useless waste of energy.

The infrastructure-based beacon-enabled Cluster-Tree topology, shown in Fig. 2.4c, combines the benefits of both topologies mentioned above such as good scalability, network synchronization, predictable performance, and energy efficient behavior, which are suited for medium-scale time-sensitive applications using battery-powered nodes. The Cluster-Tree topology is a tree-based topology such that ZC establishes the initial network while

ZRs form the branches and relay the messages. ZEDs act as leaves of the tree and do not participate in message routing. In Cluster-Tree topology, each cluster can be seen as a star topology so that one FFD (i.e., ZC or ZR) is chosen as a cluster-head that handle all the transmissions within the cluster. Thus, each cluster is composed of its cluster-head and the set of child nodes (i.e., nodes associated with given cluster-head) and, consequently, each ZR belongs to two clusters, once as a child node and once as a cluster-head. The ZC belongs only to one cluster. In contrast to the star topology, the Cluster-Tree topology supports good scalability since ZRs can grow the network beyond the initial network established by the ZC. In contrast to mesh topology, the communication is deterministic since each node only interacts with its parent and/or child nodes. Hence, there is a unique routing path between any pair of nodes within the Cluster-Tree topology, i.e., deterministic routing protocol, and therefore; the end-to-end connectivity between nodes is predictable. The deterministic routing protocol together with the contention-free medium access (GTS), as supported by beacon-enabled Cluster-Tree, ensures predictable network performance, resource bounds, and time-efficient multi-hop communications. Also, due to the synchronous behavior though emitting beacon frames by ZC and ZRs, the nodes know their allocated GTS time-slots in advance, and consequently, each node can save its energy by entering the power-saving mode when it is not engaged in any activity.

The drawback of the Cluster-Tree topology, compared to mesh network, is its less flexibility since it relies on the pre-deployed infrastructure. Moreover, even though the IEEE 802.15.4 standards and ZigBee specifications admit the formation of the Cluster-Tree network, none of them impose any algorithm or methodology to create or organize it. Thus, the Cluster-Tree topology expresses several challenging and open research issues in this area. In particular, the Cluster-Tree network needs specific algorithms to correctly design the parameters that regulate beacon and data transmission in order to achieve a good network capacity. In other words, the behavior of the whole Cluster-Tree network strongly depends on the setting of the parameters such as SO and BO. For example, if SO = BO there will be no inactive portion meaning that the nodes cannot enter into the power-saving mode. On the other hand, if SO is set too low, (and so does the duty-cycle), the data rate has to be decreased in order to fit the transmitted data within the superframe. In addition to parameters tunning, the Cluster-Tree network requires precise cluster scheduling algorithms that avoid collisions among the clusters and guarantee particular set of OoS properties. The design of such algorithms and methodologies are addressed in this thesis.

Chapter 3

Optimized TDMA Scheduling Algorithm for Single-Collision Domain ZigBee Cluster-Tree Topology

This chapter assumes a beacon-enabled Cluster-Tree network, Fig. 3.1, that has already been set up (i.e., each node knows its parent and child nodes using the ZigBee tree addressing scheme [9]). All nodes may have sensing and/or actuating capabilities; therefore they can be sources and/or sinks of data flows. The traffic is organized into a set of time-bounded multi-hops data flows (each given by parameters such as source nodes, sink node, and end-to-end deadline given in time units). The data flows parameters are known during the network configuration time. The assumption of time-bounded data flows supports applications with stringent timeliness requirement. Each data flow traverses the unique path within the Cluster-Tree topology cluster by cluster until reaching the sink node. Therefore, the data flows might have opposite directions to each other which supports industrial control application when the data flows are in both directions (i.e., from and to the field devices). For example f_2 has an opposite direction to f_3 while traversing from R_1 to R_3 in Fig. 3.1.

In Cluster-Tree topology, the cluster life cycle is periodic and each cluster is active only once within the period [9]. When the cluster is active, the cluster-head exchanges the data with its child nodes using GTSs time-slots in order to guarantee real time transmission. The neighboring clusters are in a collision and are not allowed to be active simultaneously. Thus, the key problem to solve is to find a collision-free TDMA cluster schedule which specifies the GTS time-slots allocation and the precise time at which each cluster is active while meeting the end-to-end deadlines of the data flows. Moreover, to support applications with stringent reliability demands, the acknowledgment and re-transmission mechanism needs to be considered while constructing the schedule. Furthermore, since wireless nodes are usually battery-powered, the energy efficiency of the schedule is a problem of paramount importance in order to maximize the lifetime of the network. Thus, the objective is to minimize the energy consumption of the nodes by maximizing the time when the nodes remains in power saving mode.

The collision-free TDMA cluster scheduling problem, regardless of the timeliness constraints of the data flows, is an $\mathcal{NP}$-hard problem [22]. The

Table 3.1: The user-defined parameters of the data flows from Fig. 3.1 (ptu = processing time unit).

flow ID	*source(s)*	*sink*	*e2eDeadline*		*reqPeriod*	*sampleSize*	*sampleACK*
q	(α_{f_q})	(β_{f_q})	[s]	[ptu]	[s]	[bit]	
1	1	12	1.5	1563	1	64	0
2	11	13	2	2083	2	16	1
3	14	15	2	2083	1	16	1
4	16	1	2	2083	2	64	0

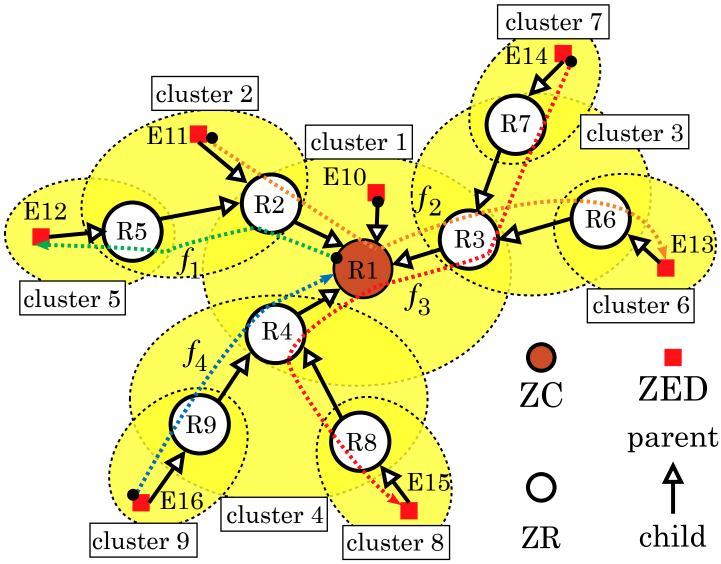


Figure 3.1: Cluster-Tree topology with 4 time bounded data flows.

proof is based on the reduction of the graph coloring problem, a well known $\mathcal{NP}$-hard problem, to the collision-free TDMA scheduling problem. To cope with the problem complexity, we simplified the problem by considering single-collision domain Cluster-Tree topology (i.e., one cluster, at most, can be active at any given time). Furthermore, we propose an elegant approach, that expresses the end-to-end deadline, given in time units, as a maximum number of crossed periods (i.e., the maximum integer when multiplied by the length of the schedule period, the result is less than or equal to the end-to-end deadline). Therefore, a collision-free TDMA schedule that meets the maximum number of crossed periods of each data flow also meets the end-to-end deadline for each data flow. The collision-free TDMA cluster scheduling problem in case of single-collision domain Cluster-Tree topology and data flows constrained by the maximum number of crossed periods can be solved in polynomial time. Therefore, we propose an exact algorithm to obtain the collision-free TDMA cluster schedule. The algorithm is based on graph theory algorithms such as shortest path and topological ordering algorithms [17].

3.1 Related Work

Energy efficiency is an important requirement for WSNs in order to maximize the lifetime of the network. The major sources of energy waste in WSNs are collisions, overhearing and idle listening [50]. We eliminate those sources of energy waste by using a collision-free TDMA scheduling algorithm that utilizes the dedicated GTS mechanism.

Koubaa et al. [32] have proposed an algorithm for collision-free beacon/superframe scheduling in single-collision domain IEEE 802.15.4/ZigBee Cluster-Tree networks, using the time division approach. The focus of the work is on the feasibility of the periodic schedule, with the goal of a fair allocated bandwidth rather than low latency.

The authors in [52] suggest a GTS allocation algorithm for periodic real-time messages in a star topology. The algorithm determines the standard specific parameters for the network and a GTS descriptor to meet the timing constraints. However, the GTS information has to be broadcasted at the beginning of each beacon interval, which increases the network power consumption. In [18], the authors propose an extension to IEEE 802.14.4 to overcome its limitation related to the number of possible allocated GTSs in one superframe. The algorithm allows more than seven periodic nodes to be simultaneously configured to one coordinator and real time transmission can still be guaranteed for each periodic node. The algorithm is based on a Window Scheduling Algorithm (WSA) [27] and improves the bandwidth utilization and the energy efficiency. However, no cluster scheduling algorithm is addressed in [52] or [18].

In [42], the authors present a solution to change the resource allocation of the Cluster-Tree on the fly. This solution is directed at applications which need to deliver data to the root of the tree. The solution is not very effective in the case of simultaneous data flows with opposite directions. In [13], a multi-cast mechanism in ZigBee Cluster-Tree WSNs was proposed. However, no time-bounded data flows were assumed.

The authors in [25] introduced a cluster and a GTS scheduling mechanism for a multi-hop Cluster-Tree WSN minimizing the energy consumption. The scheduling problem is $\mathcal{NP}$-hard while assuming multiple collision domains and precise end-to-end deadlines for the data flows. In order to find the schedule, the authors used ILP to solve small size instances. In our work, we propose two simplifications to the problem by expressing the end-to-end deadlines of the data flows in terms of the maximum number of crossed periods and by assuming single-collision domain Cluster-Tree topology. These simplifications enable us to solve the cluster scheduling problem in polynomial time.

3.2 Chapter Outline and Contribution

This chapter provides the following original contributions:

1. Simplifying the collision-free TDMA cluster scheduling problem by assuming single-collision domain Cluster-Tree topology (i.e. at most, one cluster can be active at any given time) and by expressing the precise end-to-end deadline, given in time units, into the maximum number of crossed periods. Both assumptions lead to polynomial time complexity instead of $\mathcal{NP}$-hard complexity.
2. Proposing and implementing an optimal polynomial Time Division Multiple Access for single-collision domain (TDMA^{scd}) cluster scheduling algorithm that is based on graph theory algorithms such as shortest path and topological ordering algorithms. The algorithm ensures collision avoidance, energy efficiency, timeliness, and reliability QoS properties.
3. Solving large size instances in a short time.
4. Simulation scenarios are accomplished in Opnet Modeler 17.5 to demonstrate the impact of the number of re-transmissions on the network reliability and timeliness of the data flows. Also, the energy consumption as a function to the number of re-transmissions and the duty cycle of the nodes is demonstrated.

The rest of this chapter is organized as follows: First, we provide a generic system model, encompassing the Cluster-Tree topology, the data flow model and the cyclic behavior of the periodic schedule (Sec. 3.3). In Sec. 3.4, a solution to the problem is presented and explained. In Sec. 3.4.1, the method for determining the superframe duration for each cluster is presented. Then, in Sec. 3.4.2, we illustrate the constraint model for the cosidered cluster scheduling problem. In Sec. 3.4.3, the transformation of the cluster scheduling problem to the shortest path problem is described in details so that the Partial Order of Cluster Activations (POCA) graph is constructed. In Sec. 3.4.4, POCA graph is realized as a collision-free TDMA cluster schedule. Sec. 3.5 and Sec. 3.6 show our experimental and simulation results, respectively. Finally, we draw the conclusion in Sec. 3.7.

3.3 System Model

We consider a static deployment of wireless nodes which defines the physical topology of the Cluster-Tree where each pair of connected nodes can use the shared bidirectional wireless link for the data transmission. The logical topology defines the parent-child relationship between each pair of connected nodes [9, 6].

3.3.1 Cluster-Tree Topology Model

In this chapter, we consider a Cluster-Tree topology with n nodes and m clusters where $m < n$ (for example, the Cluster-Tree topology shown in Fig. 3.1 consists of $n = 16$ nodes and $m = 9$ clusters). Fore more details about ZigBee Cluster-Tree topology, please see Chapter 2, Sec. 2.4. Let $parent_i$ be the parent node of node i, $Child_i$ be the set of the child nodes of node i, and L be the set of the leaf nodes (i.e., $L = \{i = 1 \ldots n : Child_i = \emptyset\}$). Hence the notation C_i refers to the set of nodes within the cluster C_i for which node i is the cluster-head (i.e., $C_i = \{i\} \cup Child_i$: $i \notin L$).

3.3.2 Data Flow Model

The traffics within the Cluster-Tree topology are organized into data flows with user defined parameters as shown in Tab. 3.1 . Each data flow f_q may have more than one source node but exactly one sink node (see Fig. 3.1 where 4 data flows are illustrated as dashed directed lines such that each data flow has one source node). The source node of data flow f_q, denoted by α_{f_q}, periodically measures a sensed value with a given size and required period denoted by $sampleSize_{f_q}$ and $reqPeriod_{f_q}$ respectively and reports it to the sink node β_{f_q}. The $reqPeriod_{f_q}$ is a given parameter that is associated with data flow f_q and indicates the upper bound of the time interval between two consecutive measurements performed by the source node of the data flow.

To support applications with real time demands, each data flow f_q is constrained by the end-to-end deadline, denoted by $e2eDeadline_{f_q}$, and given in time units. The $e2eDeadline_{f_q}$ specifies the maximum allowed elapsed time between the instant when the source node α_{f_q} sends the packet to the time instant when the sink node β_{f_q} receives the packet (see also Tab. 3.1 where 1 ptu $= aBaseSuperFrameDuration/16 = 0.96$ ms). Moreover, to support applications with stringent reliability demands, the acknowledgment and re-transmission mechanism might be requested. Thus, the $sampleACK_{f_q}$ for data flow f_q determines whether the acknowledgment and re-transmission mechanism is enabled or not.

Since multi-hop communication is deterministic in the Cluster-Tree

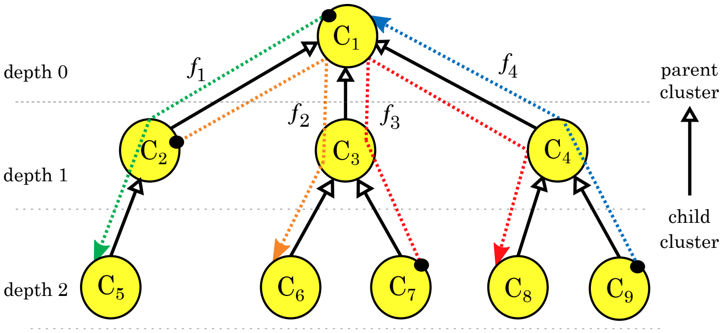


Figure 3.2: Cluster-graph that shows the source and sink clusters of each data flow in Fig. 3.1.

topology, the packets for each data flow are forwarded cluster by cluster following the unique routing path from the source cluster to the sink cluster. The source and sink clusters of f_q are determined by Def. 3.3.1 and Def. 3.3.2.

Definition 3.3.1. For each f_q with a source node $i = \alpha_{f_q}$, C_i is the source cluster if the first hop of f_q is a *parent-child* hop otherwise, $C_j : j = parent_i$ is the source cluster.

Definition 3.3.2. For each f_q with a sink node $i = \beta_{f_q}$, C_i is the sink cluster if the last hop of the f_q is a *child-parent* hop otherwise, $C_j : j = parent_i$ is the sink cluster.

In the thesis, we denote $C_{src_{f_q}}$ and $C_{sink_{f_q}}$ to be the source and the sink clusters of f_q where src_{f_q} and $sink_{f_q}$ nodes are the cluster-heads of the source and sink clusters respectively. The cluster graph that illustrates the set of clusters in addition to the source and sink clusters of each data flow as given by Def. 3.3.1 and Def. 3.3.2 for the Cluster-Tree topology shown in Fig. 3.1, is depicted in Fig. 3.2. Moreover, the depth of each cluster C_i, denoted by $depth_i$ is shown in Fig. 3.2. The solid edges represent the *parent-child* relations between the clusters so that $C_i = parent(C_j)$ if $i = parent_j$ while the dashed arcs represent the data flows. Based on the source and sink clusters of each data flow, three types of data flows are distinguished as given by Def. 3.3.3, Def. 3.3.4 and Def. 3.3.5.

Definition 3.3.3. f_q is an upstream data flow if every hop of f_q, from $C_{src_{f_q}}$ $C_{sink_{f_q}}$, is a *child-parent* hop (e.g., f_4 in Fig. 3.2).

Definition 3.3.4. f_q is a downstream data flow if every hop of f_q, from $C_{src_{f_q}}$ to $C_{sink_{f_q}}$, is a *parent-child* hop (e.g., f_1 in Fig. 3.2).

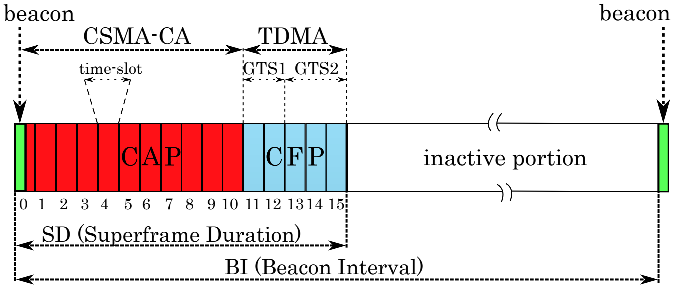


Figure 3.3: Superframe structure.

Definition 3.3.5. f_q is a bidirectional data flow where $C_{z_{f_q}}$ crossed by f_q exists so that the part from $C_{src_{f_q}}$ to $C_{z_{f_q}}$ is upstream while the part from $C_{z_{f_q}}$ to $C_{sink_{f_q}}$ is downstream (e.g., f_2 and f_3 in Fig. 3.2).

3.3.3 Cluster Life Cycle

The life cycle of the clusters is periodic and each period, corresponding to the BI, is divided into active and inactive portions (Fig. 3.3). During the inactive portion, all the nodes within the cluster go into power-saving mode to save energy. The active portion corresponding to the SD is subdivided into 16 equally sized time-slots. The beacon frame occupies the first time-slot and the remaining time-slots are partitioned into a CAP and optional CFP. The beacon frames are periodically sent by the cluster-heads to synchronize the communications within the cluster and to define the superframe structure. During the CAP, a slotted CSMA-CA protocol is used for the best-effort data delivery. Within the CFP, the cluster-head can allocate the GTSs to its child nodes for real-time transmission. The CFP supports up to 7 GTSs and each GTS may contain one or more time-slots. The values of BI and SD are defined by two parameters, the BO and the SO as shown in Eq. (2.1). In this thesis, we assume that all clusters have an equal BI, but various SD to ensure efficient bandwidth utilization.

3.3.4 Cyclic Nature of the Cluster Schedule

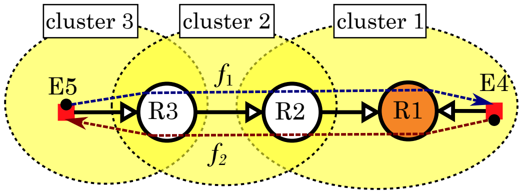
Since each cluster is active only once during the BI, and all clusters have an equal value of BI, then BI represents the length of the schedule period. This also leads to the so-called cyclic behavior of the periodic schedule [26] (i.e., there is one data flow, at least, with end-to-end delay that is longer than the period) when there are data flows with opposite directions. In such a case, the minimization of the number of periods spanned by a data flow f_i is in contradiction with the minimization of the number of periods spanned by data flow f_j when $\{f_i, f_j\}$ are in opposite direction. For example, consider a simple network with two data flows directed in opposite direction as illustrated in Fig. 3.4a. Two schedule scenarios for Fig. 3.4a are illustrated in Fig. 3.4b and Fig. 3.4c where one data communication from the source to the sink is called a wave, and the notation $f_{q,k}$ denotes the wave k of the data flow f_q. Scheduling the clusters in the sequence $C_1 \to C_2 \to C_3$, as shown in Fig. 3.4b, leads to the case at which each wave of f_2 starts and ends in the same period (i.e., spans over one period and thus, it has 0 crossed periods) while each wave of f_1 spans over three periods to reach the sink (i.e., 2 crossed periods). Scheduling the clusters in the sequence $C_3 \to C_2 \to C_1$ as shown in Fig. 3.4c leads to the case at which each wave of f_1 has 0 crossed periods while each wave of f_2 has 2 crossed periods.

Definition 3.3.6. For any cyclic schedule and for given data flows f_i and f_j so that f_i has opposite direction to f_j, then one data flow, at least, spans over multiple periods (i.e., starts in one period and ends in one of the subsequent periods). Furthermore, the e2eDelay minimization of f_i is in contradiction with the e2eDelay minimization of f_j.

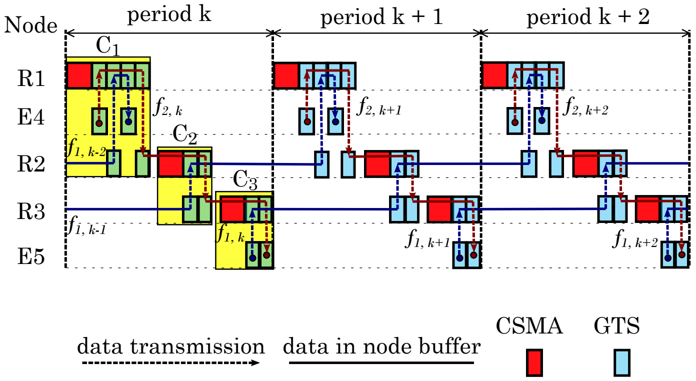
Definition 3.3.7. Let Tx denotes the GTS time-slots allocated to the child node to send the data to the cluster-head and Rx denotes the GTS time-slots allocated to the child node to receive the data from the cluster-head. Then, the transmission within the cluster (i.e., from any node $u \in C_i$ to any node $v \in C_i$ so that $u \neq v$) will commence and end within the same period if the Tx time-slots are followed by the Rx time-slots within the CFP.

Definition 3.3.8. Let the triple $(C_i, \to, C_j)$ denote that C_i is followed by C_j in the schedule (i.e., the second entry of the triple represents the precedence decision between the C_i and C_j) so that $C_i = \mathit{parent}(C_j)$. Then, transmitting the data from any node $u \in C_i$ to any node $v \in C_j$ (i.e., the transmission has the same direction to the precedence decision) will commence and end within the same period. The opposite direction transmission from any node $v \in C_j \setminus \{j\}$ to any node $u \in C_i$ will end within the subsequent period to the period at which the transmission has commenced. Notice that when $v = j$, then by Def. 3.3.7, the transmission from v to any node $u \in C_i$ commences and ends within the same period.

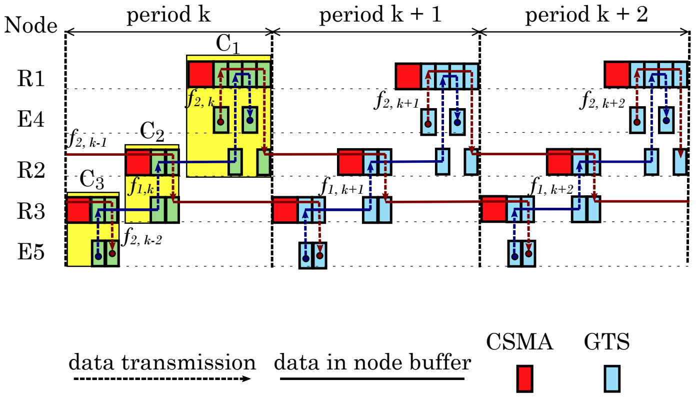
Definition 3.3.9. Let the triple $(C_i, \leftarrow, C_j)$ denote that C_j is followed by C_i in the schedule so that $C_i = \mathit{parent}(C_j)$. Then, transmitting the data from any node $v \in C_j$ to any node $u \in C_i$ will commence and end within the period while the opposite direction transmission from any node $u \in C_i$ to any node $v \in C_j \setminus \{j\}$ will end within the subsequent period. Notice that when $v = j$, then by Def. 3.3.7, the transmission from v to any node $u \in C_i$ commences and ends within the same period.



(a) Chain network with two opposite direction data flows



(b) Minimization of the end-to-end delay of data flow f_2.



(c) Minimization of the end-to-end delay of data flow f_1.

Figure 3.4: Chain network with two GTSs allocations scenarios.

Proposition 3.3.1. For any cyclic cluster schedule, if f_q commences within the period given by the interval $[x \cdot \mathrm{BI}, (x+1) \cdot \mathrm{BI})$, then the data is delivered to the sink node within the period given by the interval $[(x+\theta_{f_q}) \cdot \mathrm{BI}, (x+\theta_{f_q}+1) \cdot \mathrm{BI})$ where θ_{f_q} is the number of precedence decisions on the path of the data flow from the source cluster to the sink cluster which are directed in the opposite direction to the data flow direction.

Proof. Let f_q be a data flow with a source cluster C_u and sink cluster C_v. Given a set of precedence decisions between every two consecutive clusters on the path of f_q from the source cluster to the sink cluster, then by Def. 3.3.8 and Def. 3.3.9, every precedence decision that has an opposite direction to the data flow direction leads to a delay of one period. Since θ_{f_q} is the number of precedence decisions that have the opposite direction to the data flow direction, then if the data flow transmission commences within period k then the data reaches the sink cluster in the period $k+\theta_{f_q}$. Since period k is given by the interval $[x \cdot \mathrm{BI}, (x+1) \cdot \mathrm{BI})$, then the period $k+\theta_{f_q}$ is given by the interval $[(x+\theta_{f_q}) \cdot \mathrm{BI}, (x+\theta_{f_q}+1) \cdot \mathrm{BI})$. □

For the example illustrated in Fig. 3.4b, $\theta_{f_2} = 0$ since f_2 traverses the clusters in the order $C_1 \rightarrow C_2 \rightarrow C_3$ and the precedence decisions between every two consecutive clusters on the path of the data flow are given as $(C_1, \rightarrow, C_2)$ and $(C_2, \rightarrow, C_3)$. Since data flow f_1 traverses the cluster in the opposite direction (i.e., $C_1 \leftarrow C_2 \leftarrow C_3$), both arcs of the precedence decisions have opposite direction to f_1 direction and thus $\theta_{f_1} = 2$. In Fig. 3.4c, the precedence decisions are given as $(C_1, \leftarrow, C_2)$ and $(C_2, \leftarrow, C_3)$. Thus, $\theta_{f_1} = 0$ and $\theta_{f_2} = 2$.

The Prop. 3.3.1 enables expressing the $\mathit{e2eDeadline}_{f_q}$ in terms of the maximum number of crossed periods, denoted by h_{f_q}, where $\mathit{e2eDeadline}_{f_q} \geq \mathrm{BI}$ as follows:

$$h_{f_q} = \left\lfloor \frac{\mathit{e2eDeadline}_{f_q}}{\mathrm{BI}} \right\rfloor - 1 \tag{3.1}$$

Definition 3.3.10. The h_{f_q} is an integer value that associates the following constraint to each f_q: if f_q starts in the interval $[x \cdot \mathrm{BI}, (x+1) \cdot \mathrm{BI})$, then the data has to be delivered to the sink node during or before the interval $[(x+h_{f_q}) \cdot \mathrm{BI}, (x+h_{f_q}+1) \cdot \mathrm{BI})$.

Definition 3.3.11. The collision-free cyclic cluster schedule is feasible when $\theta_{f_q} \leq h_{f_q}$ for each data flow f_q.

For the example shown in Fig. 3.1 and by assuming $\mathrm{BI} = 1024$ ptu, then by applying Eq. (3.1) on the *e2eDeadline* given in Tab. 3.1, we get: $h_{f_1} = 0$, $h_{f_2} = 1$, $h_{f_3} = 1$ and $h_{f_4} = 1$. Let us assume that the superframe duration of the clusters given in ptu are $[32, 16, 16, 16, 16, 16, 16, 16, 16]$, then one possible collision-free cluster schedule that meets h_{f_q} for each data flow f_q is shown in Fig. 3.5. Since in this chapter, we assume single-collision domain Cluster-Tree topology, then at most, one cluster is active at any given time. Based on the schedule in Fig. 3.5, $\theta_{f_1} = 0$, $\theta_{f_2} = 1$, $\theta_{f_3} = 1$ and $\theta_{f_4} = 1$. Thus, the feasibility condition as given by Def. 3.3.11 holds.

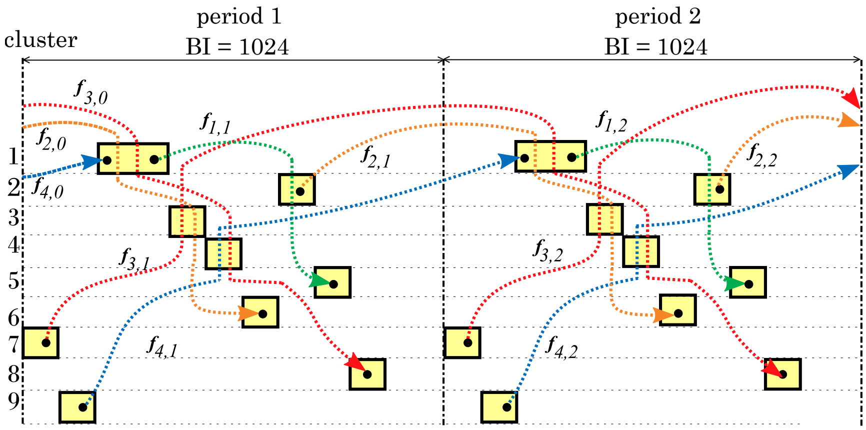


Figure 3.5: Feasible cyclic schedule for the example shown in Fig. 3.1.

3.4 TDMA$^{\text{scd}}$ Scheduling Algorithm

The fundamental problem to solve in this chapter is to find a periodic and collision-free TDMA cluster schedule such that each data flow reaches its sink cluster within the specified end-to-end deadline. Hence, it is necessary to specify at which time each cluster is active within the schedule period together with the GTSs allocation such that $\theta_{f_q} \leq h_{f_q}$ for each data flow f_q. Since, in this chapter, we consider single-collision domain Cluster-Tree topology, then at most one cluster is active at any given time. Furthermore, each cluster C_i is active only once within the schedule period for a given time denoted as SD_i. The SD_i duration is relevant to the number and length of the allocated GTSs which also depend on the payload of the data flows and whether the re-transmission and acknowledgment mechanism is enabled [25]. The objective is to minimize the energy consumption of the nodes which is equivalent to the minimization of the duty-cycle of the clusters. Since each cluster is periodically active for a fixed amount of time, then the minimization of the duty-cycle of the clusters is equivalent to the maximization of BI that leads to maximizing the time at which the nodes stay in power-saving mode. However, since h_{f_q} is inversely proportional to BI (see Eq. (3.1)), then the longer the length the BI is, the harder to satisfy the h_{f_q} of each data flow f_q.

The cluster scheduling problem is constrained by: the h_{f_q} of each data flow f_q, BI_{max}: the upper bound of BI, and BI_{min}: the lower bound of BI. BI_{max} is calculated by Eq. (2.1) based on BO_{max} that is given by the shortest *reqPeriod* among all of the data flows as shown in Eq. (3.2):

$$\mathrm{BO}_{max} = \left\lfloor \log_2 \left(\frac{\min_k(reqPeriod_{f_q})}{aBaseSuperframeDuration} \right) \right\rfloor \tag{3.2}$$

Algorithm 1: The TDMA$^{\mathrm{scd}}$ algorithm.

```
BO ← $BO_{min}$
feasible ← 0
SD ← Call Alg. 2
while $BO \leq BO_{max}$ do
    (new_feasible)← solve(data_flows, BO, Cluster-Tree)
    if new_feasible then
        feasible ← new_feasible
        BO ← BO + 1
    else
        break
if feasible then
    BO ← BO − 1
```

The value of BO_{min}, calculated by Eq. (3.3), is rounded up to the nearest BO such that the resulting period, BI_{min}, is large enough to accommodate the active portion for all the clusters when assuming at most one cluster is active at any time instant.

$$\mathrm{BO}_{min} = \left\lceil \log_2 \left(\frac{\sum_{i=1}^{m} \mathrm{SD}_i}{aBaseSuperframeDuration} \right) \right\rceil \tag{3.3}$$

The desired collision-free schedule is the one with the maximum value of BI, given by BO, such that $\mathrm{BO} \in \{\mathrm{BO}_{min}, \ldots, \mathrm{BO}_{max}\}$ and $\theta_{f_q} \leq h_{f_q}$ for each data flow f_q. To solve the scheduling problem, we propose polynomial and exact Time Division Multiple Access for single-collision domain (TDMA$^{\mathrm{scd}}$) cluster scheduling algorithm.

The pseudo code of TDMA$^{\mathrm{scd}}$ is depicted in Alg. 1. Firstly, the value of BO is initialized to BO_{min}. Then, Alg. 2 is used to calculate the superframe duration for each cluster (more details in Sec. 3.4.1). The function *solve* is used to find TDMA cluster schedule such that, at most, one cluster is active at any given time. The mechanism used by function *solve* is explained in details in Sec. 3.4.2, Sec. 3.4.3 and Sec. 3.4.4. If such a schedule exists, the value of BO is increased by 1 as long as it does not reach BO_{max}. This procedure is repeated till BO reaches BO_{max} or no feasible cluster schedule that meets the h_{f_q} for each data flow f_q exists. Recall that, when the value of BO is increased, the value of h_{f_q} is decreased as given by Eq. (3.1). At the end, BO will be equal to the largest value satisfying the feasibility condition as given by Def. 3.3.11.

3.4.1 Duration of the Cluster's Active Portion

The duration of SD is related to the number and length of each allocated GTS. Each GTS includes in addition to the effective data, the Inter-frame Spacing (IFS), eventual acknowledgment and re-transmissions. IFS separates consecutive frames and it is equal to Short Inter-Frame Spacing (SIFS) or Long Inter-Frame Spacing (LIFS) according to the length of MAC frame (see Fig. 3.6). In the case of the acknowledged transmissions (i.e., *sample_ack* = 1) the sender waits for the corresponding acknowledgment frame for at most a *macAckWaitDuration* (*macAWD*) [1]. If an acknowledgment frame is received within the *macAckWaitDuration*, the transmission is considered successful. Otherwise, the data transmission and waiting for the acknowledgment are repeated up to a maximum of *macMaxFrameRetries* (*macMFR*) times [1]. If an acknowledgment frame is not received after *macMaxFrameRetries* re-transmissions, the transmission is considered failed. The duration of a GTS required for the whole data transmission (data frame, IFS, eventual acknowledgment and re-transmissions) is expressed as:

$$\begin{aligned} \varphi &= \begin{pmatrix} frm\_size_i/rate \quad + \\ macAWD \cdot sample\_ack_i \end{pmatrix} + \Delta IFS \\ T_{GTS} &= \sum_{i=1}^{e} (macMFR \cdot sample\_ack_i + 1) \cdot \varphi \end{aligned} \tag{3.4}$$

where frm_size is the size of the transmitted frame including the data payload, MAC and PHY headers; the *rate* is the data rate which is equal to 250 kbps; ΔIFS is equal to SIFS or LIFS depending on the length of MAC frame; and e is the number of data flows in the transmit or receive direction belonging to a given child node. The number of allocated time-slots for a given GTS is then equal to:

$$N_{GTS} = \left\lceil \frac{T_{GTS}}{TS} \right\rceil \tag{3.5}$$

where TS is the duration of a time-slot and is equal to SD/16. The number of time-slots, N_{GTS}, is calculated for each allocated GTS in a given SD. The remaining time-slots of the SD are utilized for the best-effort traffic within the CAP. The allocated GTSs cannot reduce the length of the CAP to less than *aMinCAPLength* [1].

The SD calculation algorithm is presented in Alg. 2. The value of SO_i for each cluster C_i is computed iteratively starting from $\text{SO}_i = 0$. If the number of time-slots required for all allocated GTSs in a given SD_i is greater than $16 - \lceil aMinCAPLength/TS \rceil$, then SO_i is increased by 1 and the length of each GTS is recalculated by Eq. (3.5). This procedure is repeated until all allocated GTSs fit into SD_i.

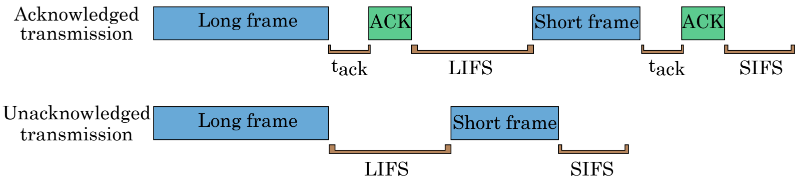


Figure 3.6: The Inter-Frame Spacing.

Algorithm 2: The calculation of the Superframe duration.

```
for each $C_i$ do
    $\mathrm{SO}_i \leftarrow -1$
    repeat
        for each child node $j$ of cluster $i$ do
            calculate $N^T_{GTS,j}$ for all data flows in transmit direction
            calculate $N^R_{GTS,j}$ for all data flows in receive direction
        $\mathrm{SO}_i \leftarrow \mathrm{SO}_i + 1$
    until $\sum_j N^T_{GTS,j} + \sum_j N^R_{GTS,j} \leq 16 - \lceil aMinCAPLength/TS \rceil$
```

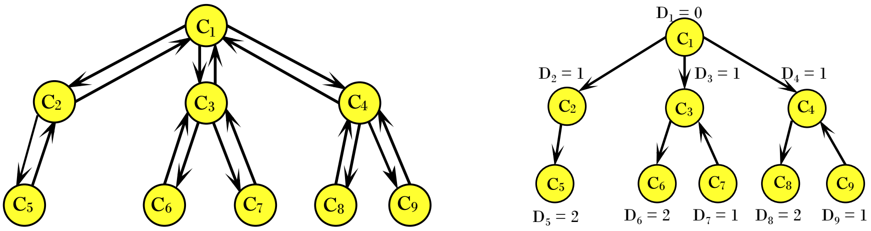
3.4.2 Modeling the Deadline Constraints of the Data Flows

As explained in Prop. 3.3.1 and Def. 3.3.11, the precedence decisions between every two consecutive clusters on the path of f_q, as will be realized by the cluster schedule, is the key problem to be solved so that $\theta_{f_q} \leq h_{f_q}$ of each f_q. Tab. 3.2 illustrates three different precedence decisions scenarios for the example shown in Fig. 3.1. The value of $\theta_{f_q} : q = 1 \ldots 4$ is calculated based on Prop. 3.3.1. The first scenario illustrates the case at which each parent cluster is followed by its child clusters in the schedule. The second scenario illustrates the case at which the child cluster is followed by its parent cluster. Notice that, by Def. 3.3.11, both scenarios lead to infeasible schedule since $\theta_{f_3} = 2 > h_{f_3} = 1$. On the other hand, the feasibility condition as defined by Def. 3.3.11 holds for the precedence decisions as given by the third scenario, which are identical to the precedence decisions as presented in Fig. 3.5.

The precedence decisions determination problem can be represented as a directed graph $G(V, E)$ where V is the set of clusters while E is the set of the directed edges that represent the potential precedence decision between each cluster and its child and parent clusters. Hence, for every two clusters C_i and C_j, where $C_i = parent(C_j)$, edge $e(C_i, C_j) \in E$ is a forward edge ($\rightarrow$) while edge $e(C_j, C_i) \in E$ is a backward edge ($\leftarrow$). Then for every pair of edges $\{e(C_i, C_j), e(C_j, C_i)\}$, one edge has to be removed so that $\theta_{f_q} \leq h_{f_q}$ holds for every f_q. The remaining edges represent the desired precedence decisions.

Table 3.2: Three scenarios illustrating the impact of the precedence decisions on the θ_{f_q} for each data flow f_q in Fig. 3.1.

scenario	θ_{f_1}	θ_{f_2}	θ_{f_3}	θ_{f_4}
$\{(C_1,\rightarrow,C_2),(C_1,\rightarrow,C_3),(C_1,\rightarrow,C_4),(C_2,\rightarrow,C_5),$ $(C_3,\rightarrow,C_6),(C_3,\rightarrow,C_7),(C_4,\rightarrow,C_8),(C_4,\rightarrow,C_9)\}$	0	1	2	2
$\{(C_1,\leftarrow,C_2),(C_1,\leftarrow,C_3),(C_1,\leftarrow,C_4),(C_2,\leftarrow,C_5),$ $(C_3,\leftarrow,C_6),(C_3,\leftarrow,C_7),(C_4,\leftarrow,C_8),(C_4,\leftarrow,C_9)\}$	2	2	2	0
$\{(C_1,\rightarrow,C_2),(C_1,\rightarrow,C_3),(C_1,\rightarrow,C_4),(C_2,\rightarrow,C_5),$ $(C_3,\rightarrow,C_6),(C_3,\leftarrow,C_7),(C_4,\rightarrow,C_8),(C_4,\leftarrow,C_9)\}$	0	1	1	1



(a) Graph representation of precedence decisions.

(b) POCA graph.

Figure 3.7: Precedence decisions determination.

Fig. 3.7a depicts the graph G for the problem instance example as shown in Fig. 3.2 while Fig. 3.7b illustrates one possible feasible precedence decisions where the labels next to each cluster C_i as denoted by $D_i \geq 0$ represent the number of forward edges from C_1 to C_i. For example, $D_9 = 1$ since one forward edge from C_1 to C_9 exists, namely $e(C_1, C_4)$. Moreover, $\theta_{f_1} = 0$ since both edges $e(C_1, C_2)$ and $e(C_2, C_5)$ have the same direction to data flow f_1. In similar manner $\theta_{f_2} = 1$ since one edge, namely $e(C_1, C_2)$ has an opposite direction to the direction of data flow f_2. In this chapter, we refer to the graph shown in Fig. 3.7b by the POCA graph.

To solve the precedence decision determination problem, it is sufficient to determine the value of D_i that is associated with each C_i. The D_i value can be calculated using ILP that is presented in Fig. 3.8. In this section, we describe step by step the set of constraints in our model where D_i is the only decision variable so that the following constraints, the topological constraints, must hold:

$$\begin{aligned} D_j - D_i &\leq 1 : C_i = parent(C_j) \\ D_i - D_j &\leq 0 : C_i = parent(C_j) \end{aligned} \tag{3.6}$$

Let $C_{src_{f_q}}$ and $C_{sink_{f_q}}$ stand for the source and the sink clusters of f_q, respectively. Then for the upstream f_q, and by Prop. 3.3.1 and Def. 3.3.11, θ_{f_q} is given by the number of forward edges from cluster $C_{src_{f_q}}$ to cluster $C_{sink_{f_q}}$ as follows:

$$D_j - D_i \leq 1 \,:\, C_i = parent(C_j)$$
$$D_i - D_j \leq 0 \,:\, C_i = parent(C_j) \qquad \text{(a)}$$

$$\forall f_q : C_{src_{f_q}} \text{ and } C_{sink_{f_q}} \text{ are the source and the sink clusters, respectively}$$
$$(D_{src_{f_q}} - D_{sink_{f_q}}) \leq (h_{f_q} - numDownHops(f_q)) = c_{f_q} \qquad \text{(b)}$$

$$D_i \geq 0 \,:\, i = 1 \ldots m \qquad \text{(c)}$$

Figure 3.8: The topological and data flows constraints.

$$\theta_{f_q} = D_{src_{f_q}} - D_{sink_{f_q}} = -(D_{sink_{f_q}} - D_{src_{f_q}}) \tag{3.7}$$

While for the downstream f_q, θ_{f_q} is given by the number of backward edges from cluster $C_{src_{f_q}}$ to cluster $C_{sink_{f_q}}$ as follows:

$$\theta_{f_q} = numHops(C_{src_{f_q}}, C_{sink_{f_q}}) - (D_{sink_{f_q}} - D_{src_{f_q}}) \tag{3.8}$$

Where the $numHops(C_i, C_j)$ is a function that returns the number of hops from C_i to C_j. Since all the hops of the downstream data flow are downstream hops (i.e., *parent-child* hops), then Eq. (3.8) can be rewritten as follows, where the $numDownHops(f_q)$ returns the number of downstream hops of f_q:

$$\theta_{f_q} = numDownHops(f_q) - (D_{sink_{f_q}} - D_{src_{f_q}}) \tag{3.9}$$

Combining Eq. (3.7) and Eq. (3.9) together, then for any bidirectional f_q that changes its direction at $C_{z_{f_q}}$, θ_{f_q} is given as follows:

$$\begin{gathered}\theta_{f_q} = -(D_{z_{f_q}} - D_{src_{f_q}}) + numDownHops(f_q) - (D_{sink_{f_q}} - D_{z_{f_q}}) \\ \Rightarrow \theta_{f_q} = numDownHops(f_q) - (D_{sink_{f_q}} - D_{src_{f_q}})\end{gathered} \tag{3.10}$$

Since the number of downstream hops in the case of upstream f_q is 0, we can generalize the calculation of θ_{f_q} for every f_q as follows:

$$\theta_{f_q} = numDownHops(f_q) - (D_{sink_{f_q}} - D_{src_{f_q}}) \tag{3.11}$$

Since, by Def. 3.3.11, $h_{f_q} \leq \theta_{f_q}$ must hold for every feasible cluster schedule, then for each f_q the following constraint must hold:

$$\begin{gathered}numDownHops(f_q) - (D_{sink_{f_q}} - D_{src_{f_q}}) \leq h_{f_q} \\ \Rightarrow (D_{src_{f_q}} - D_{sink_{f_q}}) \leq (h_{f_q} - numDownHops(f_q)) = c_{f_q}\end{gathered} \tag{3.12}$$

where $c_{f_q} \in \mathbb{Z}$ is the cost associated with the given f_q.

The constraint model for the example presented in Fig. 3.1 is illustrated in Fig. 3.9 such that $h_{f_1} = 0$ and $h_{f_2} = h_{f_3} = h_{f_4} = 1$. Notice that

$0 \le D_j - D_i \le 1$	topological constraints
$D_1 - D_5 \le -2$	for data flow f_1
$D_2 - D_6 \le -1$	for data flow f_2
$D_7 - D_8 \le -1$	for data flow f_3
$D_9 - D_1 \le 1$	for data flow f_4

Figure 3.9: The constraints for the example presented in Fig. 3.1.

the system constraint matrix is totally unimodular. The ILP task with a totally unimodular constraint matrix and integer vector for the right hand side of the constraints can be solved in polynomial time. The polynomial time algorithm that solves the above mentioned constraints is presented in Sec. 3.4.3.

3.4.3 Cluster Partial Ordering Formulated as a Shortest Path Problem

In order to solve large size instances in a reasonable amount of time, we transform the precedence determination problem into the shortest path problem as follows:

Given the set of inequality constraints as presented in Fig. 3.8, the construction of the inequality graph $Q(V, E)$ is done such that V is the set of clusters while for each constraint $D_j - D_i \le const$, an edge is added from cluster C_i to cluster C_j and weighted by $const$. Hence, each edge $e(C_i, C_j) \in E(Q)$, weighted by $c_{i,j} \in \mathbb{Z}$, represents the constraint $D_j - D_i \le c_{i,j}$. The inequality graph, for the constraints in Fig. 3.9, is depicted in Fig. 3.10 where the dashed edges, denoted as $E^f(Q) \in E(Q)$, represent the data flow constraints, while the solid edges, denoted by $E^g(Q) \in E(Q)$, represent the topological constraints. Hence, $E(Q) = E^f(Q) \cup E^g(Q)$. The value of D_i, for each cluster C_i, equals the length of the shortest path from C_1 to C_i in Fig. 3.10. Using the Bellman-Ford shortest path algorithm, we get $D = [0,\ 1,\ 1,\ 1,\ 2,\ 2,\ 1,\ 2,\ 1]$. For example $D_7 = 1$ since the shortest path from cluster C_1 to cluster C_7 in Fig. 3.10 is given by the path composed of the following edges $e(C_1, C_9), e(C_9, C_4), e(C_4, C_8), e(C_8, C_7)$ with the sum of weights equals to 1. Based on the values of D_i of each cluster C_i, the precedence decisions can be determined by Alg. 3. Furthermore, the precedence decisions are depicted by POCA graph as shown in Fig. 3.7b such that E(POCA) represents the precedence decisions between each cluster and its child clusters. The resulting precedence decisions are identical to the ones illustrated in the third scenario in Tab. 3.2.

Since some edges in the inequality graph may have negative weights for some problem instances, a negative cycle may exist and consequently, the constraint model is infeasible.

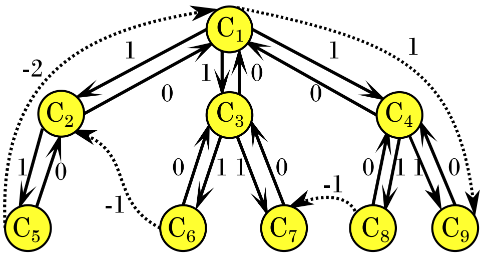


Figure 3.10: The inequality graph representation for the constraint model shown in Fig. 3.9.

Algorithm 3: The determination of the POCA graph edges.

```
E(POCA) ← $\emptyset$ // set of precedence decisions
foreach $C_i$ and $C_j$ such that $C_i = parent(C_j)$ do
    if $D_i = D_j$ then
        E(POCA) ← E(POCA) ∪ $e(C_j, C_i)$
    else
        // $D_i = D_j - 1$
        E(POCA) ← E(POCA) ∪ $e(C_i, C_j)$
```

Proposition 3.4.1. The nonexistence of a negative cycle in the inequality graph is a necessary condition for the feasibility of the precedence determination problem.

Proof. Suppose that $C_1 \to C_2 \dots C_j \to C_1$ is a negative cycle in the inequality graph. Hence, the weighted path from C_1 to C_j represents the following constraint: $D_j - D_1 \leq c_{1,j}$ and the weighted edge $e(C_j, C_1)$ represents the following constraint: $D_1 - D_j \leq c_{j,1}$. Summing both constraint leads to $0 \leq (c_{1,j} + c_{j,1})$. Since, the cycle has negative weight, then $c_{1,j} + c_{j,1} < 0$. Hence, the constraints are infeasible. □

To solve the shortest path tree problem, we use the Bellman_Ford algorithm which is able to detect the existence of cycles of negative length. Bellman_Ford runs in $O(|V||E|)$ time, where $|V|$ and $|E|$ are the number of nodes and edges respectively.

3.4.4 Topological Ordering of POCA Graph

In this section, we present the procedure that realizes POCA graph as a cluster schedule. Obviously, the POCA graph is a Directed Acyclic Graph (DAC) (i.e., directed graph with no cycles), hence there exists, at least, one topological ordering of its nodes. The topological ordering of POCA graph is an ordering of the set of the clusters such that for each $e(C_i, C_j) \in$ E(POCA), the active portion of cluster C_i must occurs before the active portion of cluster C_j within the schedule period. Since, several topological ordering might exists for given POCA graph, then every topological ordering represents one possible proper ordering for the clusters such that the resulting cluster schedule satisfies Def. 3.3.11. Since, we consider single-collision domain Cluster-Tree topology in this chapter, then at most, one cluster is active at any given time. One possible cluster schedule that is based on the topological ordering of the clusters as given in POCA graph, shown in Fig. 3.7b, is illustrated in Fig. 3.5.

3.5 Experimental Results

The experments in this chapter focus on the time complexity of the TDMA^{scd} alogirhtm. The proposed algorithm is implemented in JAVA.

The Cluster-Tree topology is constructed as follows: The routers are successively generated until the total number of the routers in the network reaches the specified number of routers, labeled by #ZRs in Tab. 3.3. Each router has 3 child end-nodes. The total number of nodes, labeled by #nodes, is shown in parentheses in the first column of Tab. 3.3.

For each Cluster-Tree topology, we generate a number of data flows as specified in the second column, labeled by #data flow. The data flows parameters such as number of sources, end-to-end deadline and required period, are specified by columns #source, e2dDeadline and reqPeriod, respectively. For simplicity, we set $\mathit{sampleSize} = 120$ and $\mathit{sampleACK} = 0$ for all data flows. Notice that the *reqPeriod* is set such that the length of the schedule period is long enough to fit the clusters within BI. For each Cluster-Tree topology and for each combination of #data flow and #source, we randomly generate a set of 30 instances and run the TDMA^{scd} scheduling algorithm. The average elapsed time, i.e., the execution time, of the algorithm to obtain the cluster schedule is shown in column elapsed time. It is clear from the results that the computation is finished in a short time even for large scale networks which proves the time efficiency of our proposed algorithm.

3.6 Simulation Study

Since the simulation is important approach to developing and evaluating the systems, we implement a simulation model in the Opnet Modeler 17.5 simulator and configured based on the TDMA^{scd}. Moreover, in this section, we show the impact of the length of BI, as given by BO, on the energy consumption of the nodes. Furthermore, since IEEE 802.15.4 supports both acknowledged and unacknowledged transmission, we also study the impact of both cases on the energy consumption, the reliability of the

Table 3.3: Time complexity of the TDMA$^{\text{scd}}$ algorithm.

#ZRs #(nodes)	#data flow	#source	e2eDeadline [s]	reqPeriod [s]	elapsed time [s]
40 (160)	2	3	12	4	0.0101
		6	12	4	0.0112
	4	3	12	4	0.0121
		6	12	4	0.0182
80 (320)	3	3	32	8	0.0724
		6	32	8	0.0773
	5	3	32	8	0.0772
		6	32	8	0.0821
150 (600)	6	3	96	16	0.6002
		6	96	16	0.6231
	10	3	96	16	0.6161
		6	96	16	0.6883
400 (1600)	12	3	256	32	0.6212
		6	256	32	0.6534
	15	3	256	32	0.7155
		6	256	32	0.8253
800 (3200)	20	3	516	64	2.3215
		6	516	64	2.7346
	25	3	516	64	2.4537
		6	516	64	3.2748

data transmission and the timeliness of the data flows. For the acknowledged transmission, the transmitter waits for a given time till it receives the acknowledgment. If waiting time elapsed while the acknowledgment is not received, the transmitter re-transmits the message. The number of re-transmissions is bounded by *macMaxFrameRetries* parameter.

3.6.1 Simulation Scenario

The simulation scenario is shown in Fig. 3.11 which is following the example presented in Fig. 3.1 (i.e., identical to the network topology and data flows). Three different icons have been utilized to differentiate between the type of nodes as illustrated in Fig. 3.11. The configuration parameters of each node are given by the TDMA$^{\text{scd}}$ algorithm. The simulation time of each run is equal to 40 min involving generation of 2396 frames of flow f_1 and f_3, and 1198 frames of flow f_2 and f_4. The number of frames is related to the simulation time and the *reqPeriod* parameter of each data flow as shown in Tab. 3.1.

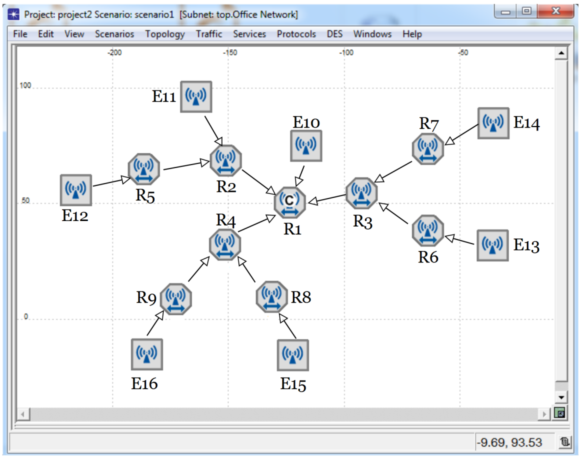

Project: project2 Scenario: scenario1 [Subnet: top.Office Network]
File Edit View Scenarios Topology Traffic Services Protocols DES Windows Help
-200
-150
-100
-50
100
50
0
E11
E10
R7
E14
E12
R5
R2
C
R1
R3
R4
R6
E13
R9
R8
E16
E15
-9.69, 93.53

Figure 3.11: The simulation scenario for the example presented in Fig. 3.1.

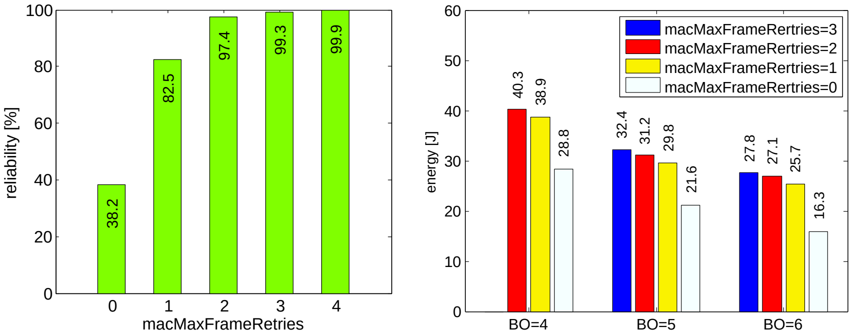
Due to different kinds of disturbances, the transmission over the frequency channel might be lost in real case WSNs. The error rate of the channel can be determined by empirically analyzing of the channel prior to the actual deployment. For the sake of simplicity, we assume that the channel error rate is fixed to 20%. Hence, the node drops the received message with the probability of 0.2.

3.6.2 Transmission Reliability

The reliability of the transmission given by the percentage of successful transmission from the source node to the sink node as a function to the number of re-transmissions is illustrated in Fig. 3.12a. Notice that the reliability of the network is directly proportional to the number of re-transmissions. Therefore, increasing the number of re-transmissions increases the reliability of the network. Moreover, given the communication error rate over the frequency channel, it is possible to specify the number of re-transmissions in order to achieve the required reliability. However, even when the communication error parameter is unknown, the number of re-transmissions must be bounded for the proper analysis and proper functionality of the TDMA^{scd} algorithm.

3.6.3 Energy Consumption

We demonstrate the energy consumption of all nodes as a function to the number of the re-transmissions and the value of BO. The energy consumption is calculated through $U \cdot I \cdot t$ where U is the voltage, I is the current



(a) Network reliability. (b) Energy consumption.

Figure 3.12: The impact of the number of re-transmissions on the network reliability and the energy consumption for given length of the schedule period.

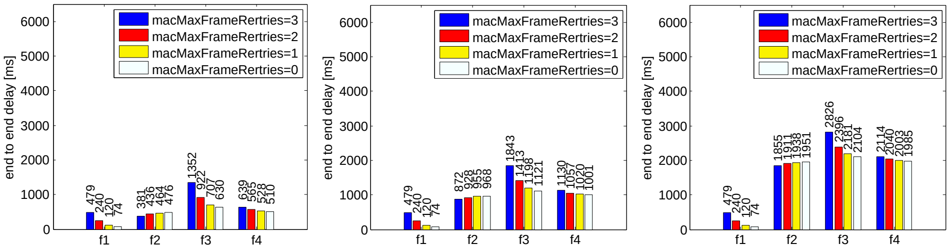
drawn and t is the execution time. The particular current drawn were given as follows: the current drawn in *receive mode* = 18.2 mA, *transmit mode* = 19.2 mA at 0 dBm, *idle mode* = 54.5 μA and *sleep mode* = 15 μA [25].

In Fig. 3.12b, we depicted the case when the maximum number of re-transmissions is given by $macMaxFrameRetries \in \{0, 1, 2, 3\}$ and BO $\in \{4, 5, 6\}$. Notice that when *macMaxFrameRetries*=3 and BO = 4, there is no feasible schedule. The energy consumption is directly proportional to the number of re-transmissions and inversely proportional to BO value. Hence, the more required reliability, the more is the energy consumption. Also, the longer the BI, the fewer energy consumptions since the nodes spends longer time into power-saving mode. Hence, a fair trade-off is required between reliability and energy efficiency.

3.6.4 Timeliness of the Data Flows

We illustrate the transmission end-to-end delay for each data flow in Fig. 3.1 as a function to both BO and the number of re-transmissions. For each case study, we apply the order of the superframe durations of the clusters as given in Fig. 3.5. However, recall that increasing the number of re-transmissions increases the superframe durations of the clusters which degrade the throughput within the network.

The results show that by increasing the number of re-transmissions, the end-to-end delay for data flows f_1, f_3 and f_4 also increases. However, the end-to-end delay of data flow f_2, for a given value of BO, decreases by increasing the number of re-transmissions. This is because of the coloration between the schedule, as shown in Fig. 3.5, and the path of data flow f_2. In other words, the superframe of cluster C_2, the source of the data flow f_2, and the superframe of cluster C_6, the sink of the data flow f_2, are



(a) BO = 5. (b) BO = 6. (c) BO = 7.

Figure 3.13: The impact of the number of re-transmissions and the length of the schedule period on the end-to-end delay of the data flows.

pushed towards the border of the schedule. However, the gap between both clusters is shortening when the superframe durations of all clusters within the network are prolonged. Furthermore, the results show that increasing the length of the schedule period, as given by the value of BO, increases the end-to-end delay for the data flows which proves the coloration between the timeliness and the energy efficiency of the network. Therefore, a fair trade-off is required.

3.7 Conclusion

This chapter addresses the collision avoidance, energy efficiency, timeliness, and reliability QoS properties. The assumptions of single-collision domain Cluster-Tree topology and the elegant approach that expresses the end-to-end deadlines of the data flows into the maximum number of crossed periods enable us to solve the addressed TDMA cluster scheduling problem in polynomial time. The algorithm is capable of solving instances of a large size and it is the core idea behind solving TDMA cluster scheduling problem for Cluster-Tree topology with multiple-collision domains as will be presented in Chapter 4. The simulation model confirms that the QoS properties are not detached from each other. For example, increasing the reliability within the network increases the energy consumption of the nodes since it prolongs the superframe duration of the clusters. Also, increasing the length of the schedule period enable more clusters to fit into the schedule period and reduces the energy consumption of the nodes. However, it has a negative impact on the timeliness QoS since it increases the end-to-end delay of the data transmissions within the network. Consequently, a fair trade-off should be found between reliability, energy consumption, and timeliness.

Chapter 4

Optimized TDMA Scheduling Algorithms for Multiple-Collision Domains ZigBee Cluster-Tree Topology

This chapter extends and completes the work presented in Chapter 3. In particular, we consider multiple-collision domains Cluster-Tree topology instead of single-collision domain Cluster-Tree as in Chapter 3. In multiple-collision domains Cluster-Tree, and in contrast to single-collision domain Cluster-Tree, several clusters might be active simultaneously (i.e., spatial reuse of the transmission medium), if they are not in collision. The spatial reuse of the transmission medium improves the bandwidth utilization and supports large scale WSNs. The clusters that are in the neighborhood are in collision and cannot be activated simultaneously [10].

In this chapter, we also follow the scenario presented in Chapter 3 such that all nodes may have sensing and/or actuating capabilities; therefore, they can be sources and/or sinks of the data flow. Each data flow is periodic and defined by a flow of messages delivered from the source node to the sink node. Since WSNs for control and monitoring applications introduce critical constraints on the delivery delay [45], we consider time-constraints on the sensed and control data (i.e., each data flow is constrained by the end-to-end deadline given in time units). Moreover, to support control applications where the data goes in both directions simultaneously (i.e., sensed data and control data from and to the field nodes), we deal with time-constrained data flows that traverse the network simultaneously in opposite directions. The data flows traverse different clusters on their routing paths from the source to the sink. Thus, each cluster-head allocates a set of GTSs periodically to its child nodes in order to send/receive data to/from its child nodes when the cluster is active. As defined by the standards [9], the lifetime behavior of the cluster is periodic. Thus, each cluster is active only once within its period for a duration given by the number and the length of the allocated GTSs. Thus, the fundamental problem to solve is to find a collision-free TDMA cluster schedule specifying, in addition to the GTSs allocation, the time at which each cluster will be active within the schedule period such that the end-to-end deadline for each data flow is met. Since the wireless nodes are usually battery-powered, the energy efficiency of the schedule is a problem of paramount importance in order to maximize the lifetime of the network.

Because each cluster is active only once within the period for a fixed time, then maximizing the length of the period is the objective to maximize the time when the clusters are in a power-saving mode. However, the larger the period, the harder it is to satisfy the end-to-end deadline for each data flow. Due to the communication error over the frequency channel, the messages might be lost or corrupted. Hence, the re-transmission and acknowledgment mechanism is adopted in order to address the network reliability within the network.

The collision-free TDMA cluster scheduling problem, while considering Cluster-Tree topology with multiple-collision domains is an $\mathcal{NP}$-hard regardless of the timeliness constraints of the data flows. The proof is based on the reduction of the graph coloring problem into a cluster scheduling problem. The difficulty of the problem increases significantly when the traffic is organized as time-constrained data flows with opposite directions. To solve the problem, we propose an exact algorithm based on ILP approach. However, since the ILP approach is impractical to solve large scale instances [25, 7, 40], we also present a novel heuristic scheduling algorithm to obtain the desired schedule in a short time even for instances with thousands of nodes. The heuristic algorithm is based on very interesting formulations of graph theory problems such as shortest path and graph coloring problems. Thus, it is efficient in both computational time (instances with thousands of nodes are solved in a short time) and solution quality (evaluated over smaller size instances while comparing it with optimal solutions obtained by ILP).

4.1 Related Work

Energy efficiency is an essential requirement for WSNs to maximize the lifetime of the network [40, 15]. The authors in [50] indicated that collisions, overhearing and idle listening are the major sources of energy waste in WSNs. We eliminate those sources of energy waste by inducing a collision-free cluster schedule and dedicated allocations of the contention free GTSs.

Many researchers tackled different scheduling problems for various WSN topologies. The work presented in [52] suggested a scheduling algorithm for periodic *real-time* flows in a star topology. An extension to the IEEE 802.15.4 to overcome its limitation related to the number of possible allocations of GTSs in one *superframe* was proposed in [18]. The algorithm allows more than seven periodic nodes to be simultaneously configured to one cluster-head and real-time transmission can still be guaranteed for each periodic node. However, neither [52] nor [18] propose a scheduling algorithm for WSN topology with multiple clusters in which the collision problem between the interfering clusters arises.

A cluster scheduling algorithm for a ZigBee Cluster-Tree using the time division approach was presented in [32]. The focus of the work is on the fair allocation of bandwidth among clusters rather than low latency. In [42], the authors presented a dynamic cluster scheduling algorithm for a Cluster-Tree WSN where all flows are directed to the root of the tree. The proposed cluster scheduling algorithm in [47] supports the scalability of the network through the utilization of multiple radio channels in order to allow simultaneous activation of interfering clusters. In [35], the authors proposed a set of time-slots allocation schemes in order to improve the network throughput and to avoid the network congestion, high end-to-end communication delays and discarded messages due to buffer overflows. However, neither [32, 42, 47] nor [35] propose algorithms to support time-constrained data flows that simultaneously traverse the network with opposite directions.

The work proposed in [25] addresses a similar cluster scheduling problem to the one considered in this research. The authors proposed a TDCS based on ILP for small size instances (less than a hundred nodes) with precise end-to-end deadline of each flow given in time units. The implementation details of the simulation model for the TDCS algorithm, which is done in the Opnet Modeler 15 simulator, were shown in [29]. The simulation results revealed the impact of the number of re-transmissions on the network reliability, the energy consumption, and the end-to-end communication delay in a way that improving one may degrade the others. In [8], we simplify the scheduling problem presented in [25] by assuming single-collision domain (i.e., at most, one cluster can be active at any given time) and by expressing the precise end-to-end deadline into the maximum number of periods crossed by each flow till its delivery. These two simplifications lead to the polynomial complexity of the problem which is solved optimally based on graph theory algorithms, namely the shortest path and topological ordering algorithms. In this research, we followed the assumption of expressing the precise end-to-end deadline as the maximum number of crossed periods [8], while we have considered the multiple collision domains [25]. Hence, the complexity of the problem remains an $\mathcal{NP}$-hard [22].

4.2 Chapter Contributions and Outline

This chapters provides the following original contributions:

1. A realistic model with multiple-collision domains Cluster-Tree topology and multi-hops data flows constrained by end-to-end deadlines expressed by the maximum number of crossed periods.
2. Proposing and implementing an optimal Exact Time Division Multiple

Access for multiple-collision domains (E_TDMA$^{\mathrm{mcd}}$) cluster scheduling algorithm that is based on ILP. The algorithm addresses collision avoidance, energy efficiency, timeliness and reliability QoS properties for small-size instances.

3. Proposing and implementing Heuristic Time Division Multiple Access for multiple-collision domains H_TDMA$^{\mathrm{mcd}}$ cluster scheduling algorithm that is based on a sound formulation of problems and subproblems concerning combinatorial optimization and graph theory. The algorithm addresses collision avoidance, energy efficiency, timeliness and reliability QoS properties for large-size instances with thousands of nodes.
4. The comparison with the TDCS algorithm presented in [25] and the evaluation over large-scale benchmarks are demonstrated.
5. Simulation scenarios are accomplished in Opnet Modeler 17.5 in order to demonstrate the correlation among several QoS properties such as reliability, energy efficiency, and timeliness.

The rest of the chapter is organized as follows: In Sec. 4.3, we provide a generic system model. The proposed TDMA scheduling algorithms are explained in Sec. 4.4. The Exact approach based on ILP is explained in Sec. 4.4.1 while the novel heuristic algorithm is demonstrated in Sec. 4.4.2. We demonstrate our computational results in Sec. 4.5. The simulation scenarios are presented in Sec. 4.6. We draw the conclusion in Sec. 4.7.

4.3 System Model

This chapter is built on top of Chapter 3, hence, we consider the similar system model as presented in Chapter 3. However, this chapter considers realistic model with multiple-collision domains Cluster-Tree topology instead of single-collision domain Cluster-Tree topology. Therefore, in this section, we only present the multiple-collision domains model. The Cluster-Tree topology model, the data flow model, the cluster life cycle and the cyclic nature of the cluster schedule are presented in Chapter 3, Sec. 3.3. Moreover, we also follow the same example presented in Chapter 3 such as the Cluster-Tree topology is shown in Fig. 3.1 while the data flow parameters are presented in Tab. 3.1. We also follow the elegant approach that expresses the $\mathit{e2eDeadline}_{f_q}$ for each data flow f_q as given in time units, into the maximum number of crossed periods as denoted by h_{f_q}. The h_{f_q} is calculated by Eq. (3.1).

4.3.1 Multiple Collision Domains Model

The spatial reuse of the transmission medium is crucial for large-scale WSNs where thousands of nodes are deployed in a large area [22]. Two clusters can be activated simultaneously in case they are not interfering with each other (i.e., they are not in collision). The collision domain of a cluster depends on the physical deployment of the WSN and on the transmission area and the carrier sensing area of the sensor nodes. The transmission area and the carrier sensing area of a node depend on the strength of the radio signal and heterogeneity of the environment. The receiver can receive and decode the message if it is in the transmission area of the transmitter. The node, which is in the carrier sensing area but not in the transmission area, is able to sense the transmission but cannot decode the message. Hence, two clusters C_i and C_j are in collision, if and only if, any node $u \in C_i$ is within the carrier sensing area of any node $v \in C_j$ or vise versa. When cluster C_i is active, the transmission within the cluster is enabled, while it is prohibited within any cluster C_j with a carrier sensing area that reaches any node in cluster C_i, thus, the hidden node problem is eliminated [49, 28].

Since we consider the static deployment of the nodes in relatively controlled environments (i.e., applications in industrial and home automation), the coordinates, the transmission area and the carrier sensing area of all nodes in the Cluster-Tree are known in advance. Thus, the symmetric collision matrix CD can be computed such that $CD_{i,j} = 1$ if C_i is in collision with C_j or vice versa, $CD_{i,i} = 2$ and $CD_{i,j} = 0$ when C_i and C_j are not in collision.

For the Cluster-Tree in Fig. 3.1, we assume that $CD_{4,6} = CD_{6,4} = CD_{6,9} = CD_{9,6} = CD_{7,9} = CD_{9,7} = 0$. Let us assume that $h_{f_1} = 0$, $h_{f_2} = 1$, $h_{f_3} = 1$, $h_{f_4} = 1$, and the superframe duration of the clusters given in ptu are $[32, 16, 16, 16, 16, 16, 16, 16, 16]$ and BI = 1024 ptu, then one possible collision-free TDMA cluster schedule that meets h_{f_q} for each data flow f_q is shown in Fig. 4.1 where the set of simultaneously activated clusters is $\{\{7, 9\}, \{4, 6\}\}$.

4.4 TDMA$^{\text{mcd}}$ Cluster Scheduling Algorithm

The fundamental problem to solve in this chapter is to find a periodic and collision-free TDMA cluster schedule such that each data flow reaches its sink cluster within the specified end-to-end deadline. Hence, it is necessary to specify at which time each cluster is active within the schedule period together with the GTSs allocation such that $\theta_{f_q} \leq h_{f_q}$ for each flow f_q as defined by Def. 3.3.11. Each cluster C_i is active only once within the schedule period for a given time denoted as SD$_i$. The SD$_i$ duration is relevant

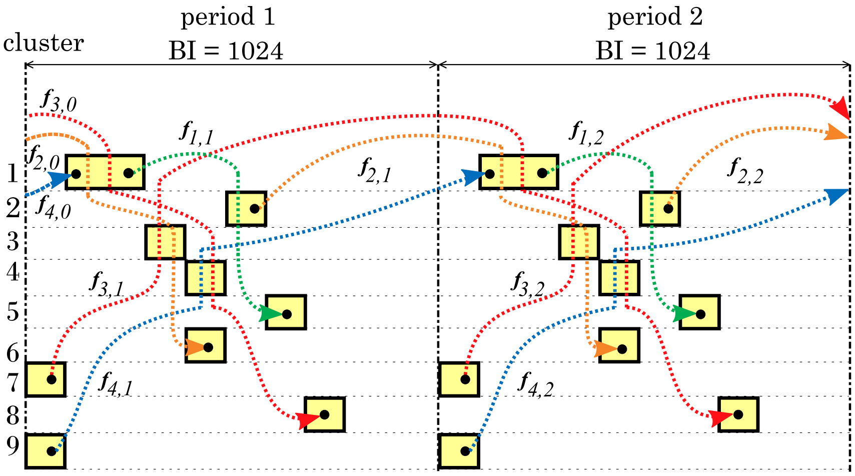


Figure 4.1: Feasible cyclic schedule for the example shown in Fig. 3.1 with the spatial reuse of the transmission medium.

to the number and length of the allocated GTSs which also depend on the payload of the data flows and whether the re-transmission and acknowledgment mechanisms are enabled [25]. The objective is to minimize the energy consumption of the nodes which is equivalent to the maximization of BI that leads to maximizing the time at which the nodes stay in power-saving mode. Since each cluster is periodically active for a fixed amount of time, then the larger the BI is, the inactive portion becomes larger, and the energy consumption of the network is reduced. However, since h_{f_q} is inversely proportional to BI (see Eq. (3.1)), then the longer the length the BI is, the harder to satisfy the h_{f_q} of each flow f_q.

The cluster scheduling problem is constrained by: the h_{f_q} of each flow f_q, BI$_{max}$: the upper bound of BI, BI$_{min}$: the lower bound of BI, and the collision matrix *CD*. BI$_{max}$ is calculated by Eq. (2.1) based on BO$_{max}$ given by the shortest *reqPeriod* among all of the flows as shown in Eq. (3.2).

BO$_{min}$ is rounded up to the nearest BO such that the resulting period, BI$_{min}$, is large enough to accommodate the active portion for all the clusters when assuming that the non interfering clusters overlap as shown in Eq. (4.1):

$$\mathrm{BO}_{min} = \left\lceil \log_2 \left(\frac{\mathrm{BI}_{min}}{aBaseSuperframeDuration} \right) \right\rceil \tag{4.1}$$

The desired collision-free schedule is the one with the maximum value of BI, given by BO, such that BO $\in \{\mathrm{BO}_{min}, \ldots, \mathrm{BO}_{max}\}$ and $\theta_{f_q} \leq h_{f_q}$ for each flow f_q.

The pseudo code of the TDMA$^{\text{mcd}}$ scheduling algorithm for Cluster-Tree topology with multiple-collision domains is depicted in Alg. 4. First,

Algorithm 4: The TDMA$^{\text{mcd}}$ algorithm.

```
BO ← $\text{BO}_{min}$
feasible ← 0
SD ← Call Alg. 2
while $BO \leq BO_{max}$ do
    (new_feasible) ← solve(data_flows, BO, Cluster-Tree)
    if new_feasible then
        feasible ← new_feasible
        BO ← BO + 1
    else
        break
if feasible then
    BO ← BO − 1
```

the value of BO is initialized to BO_{min}. Then, Alg. 2 is used to calculate the superframe duration for each cluster. The function *solve* is used to find the required cluster schedule. In case of E_TDMA$^{\text{mcd}}$, the function *solve* utilizes the ILP model as presented in Sec. 4.4.1 while in case of H_TDMA$^{\text{mcd}}$, function *solve* utilizes the algorithm illustrated in Alg. 5. If such a schedule exists, the value of BO is increased by 1 as long as it does not reach BO_{max}. This procedure is repeated till BO reaches BO_{max} or no feasible cluster schedule that meets the h_{f_q} for each data flow f_q exists. Recall that, when the value of BO is increased, the value of h_{f_q} is decreased as given by Eq. (3.1). At the end, BO will be equal to the largest value satisfying the required deadlines of the data flows.

4.4.1 E_TDMA$^{\text{mcd}}$ Scheduling Algorithm

The ILP formulation of the collision-free TDMA cluster scheduling problem for multiple-collision domains Cluster-Tree topology considering the precise *e2eDeadline* given in time units is presented in [25]. In this Chapter, we present the ILP formulation while considering the deadlines of the data flows as given as the maximum number of crossed periods. Both models are evaluated in our experiments in Sec. 4.5. In addition to the set of constraints that enforces the deadlines constraints as presented in Chapter 3, Sec. 3.4.2, the none overlapping constraints for the clusters that are in collision as given in matrix *CD* have to be considered.

The ILP formulation is shown in Fig. 4.2 such that the resulting schedule is given by $(s_1, s_2, ..., s_m)$, where $s_i \in [0, \text{BI} - \text{SD}_i]$ is the start time of the active portion of the cluster C_i in the schedule and m is the number of clusters in the given Cluster-Tree topology. The objective function (4.2a) and the constraints (4.2b) are essential in case a compact schedule is required, i.e., minimization of the makespan of the schedule. The other constraints are mainly of three types as follows:

$$
\begin{array}{lr}
\min\ Cmax & \text{(a)}\\[4pt]
\quad \forall\ i = 1 \dots m & \\
\qquad s_i - Cmax \le -\mathrm{SD}_i & \text{(b)}\\[6pt]
\quad \forall\ C_i = parent(C_j) & \\
\qquad D_j - D_i \le 1 & \\
\qquad D_i - D_j \le 0 & \text{(c)}\\[6pt]
\quad \forall f_q : C_{src_{f_q}} \text{ and } C_{sink_{f_q}} \text{ are the source and the sink clusters, respectively} & \\
\qquad (D_{src_{f_q}} - D_{sink_{f_q}}) \le (h_{f_q} - numDownHops(f_q)) = c_{f_q} & \text{(d)}\\[6pt]
\quad \forall\ C_i = parent(C_j): & \\
\qquad s_j - s_i - \mathrm{BI} \cdot (D_j - D_i) \le -\mathrm{SD}_j & \text{(e)}\\
\qquad s_i - s_j + \mathrm{BI} \cdot (D_j - D_i) \le \mathrm{BI} - \mathrm{SD}_i & \text{(f)}\\[6pt]
\quad \forall\ CD_{i,j} = 1 \ \ and \ \ i < j & \\
\qquad s_i - s_j - \mathrm{BI} \cdot y_{ij} \le -\mathrm{SD}_i & \text{(g)}\\
\qquad s_j - s_i + \mathrm{BI} \cdot y_{ij} \le \mathrm{BI} - \mathrm{SD}_j & \text{(h)}\\[4pt]
\quad where\text{: } s_i \in \langle 0, \mathrm{BI} - \mathrm{SD}_i \rangle;\ D_i \in \{0, ..., depth(C_i)\};\ y_{ij} \in \{0, 1\} &
\end{array}
$$

Figure 4.2: The constraint model.

The constraints (4.2c) and (4.2d) are the topological constraints and the data flows deadlines constraints. The constraints (4.2e) and (4.2f) are direct application to the precedence relations given by D_i and D_j such that $C_i = parent(C_j)$. Recall that D_i is a decision variable that is associated with each cluster C_i and represents the number of forward edges from C_1 to C_i such that:

1. When $D_i = D_j$, then the constraint (4.2e) is reduced to $s_j + \mathrm{SD}_j \le s_i$ which ensures that cluster C_j is followed by cluster C_i in the schedule. The constraint (4.2f) is reduced to $s_i + \mathrm{SD}_i \le s_j + \mathrm{BI}$ which is eliminated since it is always satisfied due to the domain definition of the variable s_i.
2. When $D_j = D_i + 1$, then constraint (4.2e) is reduced to $s_j + \mathrm{SD}_j \le s_i + \mathrm{BI}$ which is eliminated since it is always satisfied while constraint (4.2f) is reduced to $s_i + \mathrm{SD}_i \le s_j$ which ensures that cluster C_i is followed by cluster C_j in the schedule.

The constraints (4.2g) and (4.2h) ensure that clusters which are in collision are not overlapping in the schedule. The decision variable y_{ij} is a binary variable such that:

1. When $y_{ij} = 0$, constraint (4.2g) is reduced to $s_i + \mathrm{SD}_i \le s_j$ which ensures that cluster C_i is followed by cluster C_j in the schedule (i.e., not overlapping). While the constraint (4.2h) is eliminated since $s_j +$

$\mathrm{SD}_j \leq s_i + \mathrm{BI}$ is always satisfied due the definition domain of the variable s.

2. When $y_{ij} = 1$, constraint (4.2g) is eliminated since $s_i + \mathrm{SD}_i \leq s_j + \mathrm{BI}$ is always satisfied. While constraint (4.2h) is reduced to $s_j + \mathrm{SD}_j \leq s_i$ which ensures that cluster C_j is followed by cluster C_i in the schedule (i.e., not overlapping).

The above mentioned exact algorithm is implemented in JAVA and the Gurobi solver is used to solve the ILP model.

4.4.2 H_TDMA$^{\mathrm{mcd}}$ Scheduling Algorithm

In this section, we propose H_TDMA$^{\mathrm{mcd}}$ as a heuristic algorithm to solve the collision-free TDMA cluster scheduling problem. The aim of the heuristic algorithm is to cope with the complexity of the ILP model utilized by the E_TDMA$^{\mathrm{mcd}}$ and, consequently, to solve large size instances in a reasonable amount of time.

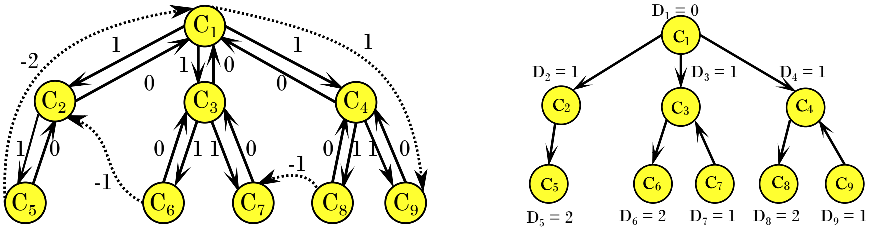
As explained in Prop. 3.3.1 and Def. 3.3.11, the precedence decisions between every two consecutive clusters on the path of each data flow f_q, as will be realized by the cluster schedule, is the key problem to be solved so that $\theta_{f_q} \leq h_{f_q}$ of each f_q. To solve the precedence decision determination problem, we use the constraint model presented in Chapter 3 (Fig. 3.8). The constraint model for the example presented in this chapter is shown in Fig. 4.3. To calculate D_i, we presented in Sec. 3.4.3 an exact polynomial centralized algorithm which consists of two parts:

1. Construction of the inequality graph $Q(V, E)$ where V is the set of clusters while for each constraint $D_j - D_i \leq \mathit{const}$, an edge is added from cluster C_i to cluster C_j and weighted by a *const*. Hence, each edge $e(C_i, C_j) \in E(Q)$, weighted by $c_{i,j} \in \mathbb{Z}$, represents the constraint $D_j - D_i \leq c_{i,j}$. The inequality graph, for the constraints in Fig. 4.3, is depicted in Fig. 4.4a where the dashed edges, denoted as $E^f(Q) \in E(Q)$, represent the data flow constraints, while the solid edges, denoted by $E^g(Q) \in E(Q)$, represent the topological constraints. Hence, $E(Q) = E^f(Q) \cup E^g(Q)$.
2. The value of D_i, for each cluster C_i, equals the length of the shortest path from C_1 to C_i in Fig. 4.4a. The precedence decisions are depicted by POCA graph as shown in Fig. 4.4b such that E(POCA) represents the precedence decisions between each cluster and its child clusters.

Alg. 5 uses the resulting POCA graph to find a collision-free TDMA cluster schedule that is compact (i.e., minimizing the length of the makespan of the schedule). The algorithm creates a directed acyclic weighted graph $T(V, E, w)$, called a task graph, such that V is the set of the clusters plus one

$0 \leq D_j - D_i \leq 1$	topological constraints
$D_1 - D_5 \leq -2$	for data flow f_1
$D_2 - D_6 \leq -1$	for data flow f_2
$D_7 - D_8 \leq -1$	for data flow f_3
$D_9 - D_1 \leq \ \ 1$	for data flow f_4

Figure 4.3: The topological and data flows constraints for the example presented in Fig. 3.1.



(a) The inequality graph.

(b) POCA graph.

Figure 4.4: Precedence decisions determination.

additional dummy cluster indexed by $m+1$ such that $\mathrm{SD}_{m+1} = 0$. Therefore, each cluster C_i, in POCA graph, is represented as a task $T_i \in V(T)$ with execution time equals to SD_i. E is the set of edges such that $e(T_i, T_j) \in E(T)$ when:

1. One data flow, at least, is crossing both clusters C_i and C_j and $e(C_i, C_j) \in \mathrm{E(POCA)}$.
2. Or cluster C_i has no successor in POCA. Then $j = m + 1$.

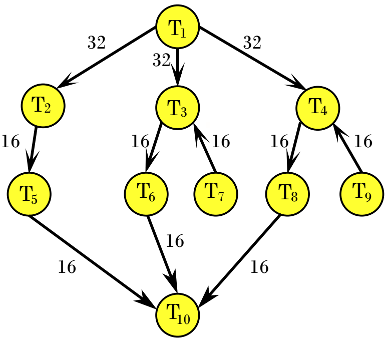


Figure 4.5: The Task graph that is based on POCA graph in Fig. 4.4b.

Each edge $e(T_i, T_j) \in E(T)$ is weighted by $w(T_i, T_j) = \mathrm{SD}_i$ (see Fig. 4.5). Then, the key problem to solve is to find the schedule for the set of tasks in T so that T_i and T_j do not overlap when $CD_{ij} = 1$.

In addition to s_i, the start time of task T_i, the algorithm uses the following variables: d_i is the length of the longest path from T_i to T_{m+1}, δ_i^+ is

Algorithm 5: The H_TDMA$^{\mathrm{mcd}}$ scheduling algorithm.

```
1  $(T, s, d, \delta^{+}, \mu) \leftarrow init(\mathrm{POCA}, CD, SD)$
2  $LB \leftarrow \max_i (d_i)$, $schedule \leftarrow \emptyset$
3  $\#notScheduledTasks \leftarrow |V(T)| - 1$
4  $feasible \leftarrow true$
5  while $\#unscheduledTasks > 0$ do
6      $T_x \leftarrow getReadyTask(T, s, d, \delta^{+}, \mu)$
7      $schedule \leftarrow addTask(T_x)$
8      for each not scheduled task $T_i$ do
9          if $e(T_x, T_i) \in E(T)$ or $CD_{i,x} = 1$ then
10             $s_i \leftarrow max\{s_i,\ s_x + SD_x\}$
11             $LB \leftarrow max\{LB,\ s_i + d_i\}$
12             if $LB > BI$ then
13                 $feasible \leftarrow false$
14                 break
15         else
16             $\mu_i \leftarrow \mu_i - 1$
17     if $\neg feasible$ then
18         break
19     $\#notScheduledTasks \leftarrow \#notScheduledTasks - 1$
```

the out degree of task T_i and μ_i is the number of clusters which are not in collision with cluster C_i, as given by the *CD* matrix, and their corresponding tasks have not been executed yet. The lower bound of the schedule length, denoted by *LB*, is given by the length of the longest path among all nodes in T to the dummy node.

To find the task schedule, the algorithm initializes $s_i = 0$ for each task T_i. Then function *getReadyTasks* returns the ready task with the highest priority, denoted by T_x. The task is classified as ready when all of its predecessor tasks are scheduled. The ready tasks are prioritized as follows: the task with the smallest value of s_i has the highest priority. If there is a tie, the highest priority task is the one with the greatest value of δ_i^{+}. If two tasks have the same δ_i^{+}, then the one with the smallest value of d_i is the highest priority task. Again, if a tie exists, then the one with the smallest value of μ_i is the task with the highest priority. Function *addTask* schedules T_x at s_x for a duration given by SD_x. For each unscheduled task T_i, that is successor to or in collision with T_x, the variable s_i is updated as shown in line 10, and consequently, *LB* is updated as shown in line 11. Selecting the highest priority ready task, scheduling and updating of the start time of the tasks are repeated till all of the tasks are scheduled or the *LB* value exceeds BI. The latter case indicates that no feasible solution was found by the algorithm since the tasks cannot fits into the schedule period as given by BI.

4.5 Experimental Results

In our experiments, we compare the proposed algorithms E_TDMA$^{\mathrm{mcd}}$ and H_TDMA$^{\mathrm{mcd}}$ with the exact algorithm TDCS presented in [25]. The TDCS algorithm considers precise end-to-end deadline as given in time units. All algorithms are implemented in Java and the exact algorithms use a Gurobi solver to solve the ILP model.

4.5.1 Benchmarks Settings

Each problem instance is constructed as follows. The ZC is placed in the center of the square of size (2000×2000) m^2. The ZRs and ZEDs are distributed such that all routers and end nodes join the tree. The transmission and the carrier sensing range for each node are given by 25 m and 40 m respectively. Each router may have up to 3 ZRs child nodes and exactly 3 child ZEDs. The total number of routers #ZRs and the total number of nodes (#nodes) in the Cluster-Tree are shown in the first column of Tab. 4.1. For each number of routers, we generate 30 different Cluster-Tree topologies.

For each resulting Cluster-Tree topology, we generate a number of data flows with a given number of sources denoted by #flows and (#src) respectively in the second column. For each data flow, we set *sampleSize* = 64 bits; and *sampleACK* = 0. The *reqPeriod* and *e2eDeadline* for the generated flows are set as follows: First, the BI_{min} is found by setting the *e2eDeadline* to infinity. Then, BO_{min} is calculated as shown in Eq. (4.1). Then *reqPeriod* is set to BI_{min}, thus, by Eq. (3.2), $\mathrm{BO}_{max} = \mathrm{BO}_{min}$. Then, the *e2eDeadline* is set to a tight value and increased for successive benchmarks where each benchmark consists of 30 instances. Thus, for the successive benchmarks with the equal *reqPeriod*, the *e2eDeadline* is increased by an integer number of the *reqPeriod*. Moreover, when all generated instances are solved by all algorithms with $\mathrm{BO} = \mathrm{BO}_{max}$, the *reqPeriod* is increased such that BO_{max} is increased by one (see columns 3–5 in Tab. 4.1 for benchmarks with 20 and 60 routers).

4.5.2 Computation Time of the Algorithms

The average computation time required by the corresponding algorithm to compute the feasible schedule and the compact schedule are depicted in the t_{feas} and t_{com} columns in Tab. 4.1 respectively. The compact schedule corresponds to the minimization of the schedule's makespan.

For the exact algorithms, we ignore the ILP model construction time and we only measure the execution time of the Gurobi solver to find the required schedule. The solution times, which exceed the time limit of 600 s, are not

Table 4.1: Time computation and solution quality.

Problem Instance					TDCS				TDMA^{mcd}							
#ZRs	#flows	e2eDeadline	reqPeriod	BO$_{max}$	Exact				E_TDMA^{mcd}				H_TDMA^{mcd}			
					t_{feas}	t_{com}	$\lvert F_{BO_{max}} \rvert$	$\lvert F_{\leq BO_{max}} \rvert$	t_{feas}	t_{com}	$\lvert F_{BO_{max}} \rvert$	$\lvert F_{\leq BO_{max}} \rvert$	t_{feas}	t_{com}	$\lvert F_{BO_{max}} \rvert$	$\lvert F_{\leq BO_{max}} \rvert$
(#nodes)	(#src)	[s]	[s]		[s]	[s]			[s]	[s]			[s]	[s]		
20 (80)	6 (4)	∞	0.25	4	0.029	200.6	1	1	0.014	148	1	1	0.0062	0.0068	1	1
		1	0.5	5	0.056	163.5 (16)	22	22	0.026	115.6 (9)	9	10	0.0076	0.009	9	10
		1.5	0.5	5	0.031	245 (25)	30	30	0.020	333 (27)	29	29	0.0034	0.0034	29	29
		2	0.5	5	0.0247	80.4 (27)	30	30	0.018	– (30)	30	30	0.0026	0.0043	30	30
		2	1	6	0.066	250.6 (24)	22	30	0.020	96.9 (29)	9	30	0.0027	0.0033	9	30
		3	1	6	0.047	145.3 (26)	30	30	0.020	235.4 (28)	29	30	0.003	0.0043	29	30
		4	1	6	0.034	22.7 (27)	30	30	0.0174	223.6 (29)	30	30	0.0023	0.0033	30	30
60 (240)	6 (4)	2	1	6	4.8	– (2)	2	2	–	–	0	0	–	–	0	0
		3	1	6	1.816	– (18)	18	18	0.274	– (4)	4	4	0.02	0.021	4	4
		4	1	6	0.491	– (28)	28	28	0.20	– (22)	22	22	0.011	0.0089	22	22
		5	1	6	0.21	– (28)	28	28	0.20	– (28)	28	28	0.0071	0.0089	28	28
		4	2	7	0.813	– (28)	2	28	0.204	– (22)	0	22	0.0063	0.0074	0	22
		6	2	7	2.761	– (30)	21	30	0.186	– (28)	4	28	0.0055	0.0061	4	28
		8	2	7	0.72	– (30)	30	30	0.26	– (30)	24	30	0.0064	0.008	24	30
		10	2	7	0.72	– (30)	30	30	0.26	– (30)	30	30	0.0061	0.0063	30	30
100 (400)	12 (4)	16	4	8	13.86	–	30	30	1.45	–	13	28	0.015	0.018	13	28
		20	4	8	2.132	–	30	30	1.985	–	30	30	0.018	0.019	30	30
		32	8	9	15.281	–	30	30	1.459	–	13	30	0.012	0.019	13	30
		40	8	9	3.95	–	30	30	2.667	–	30	30	0.014	0.018	30	30
250 (1000)	20 (4)	20	4	8	158.2 (9)	–	7	7	103.49	–	16	16	0.213	0.219	16	16
		24	4	8	53.29 (4)	–	26	26	32.07	–	30	30	0.201	0.204	30	30
		28	4	8	41.30	–	30	30	25.35	–	30	30	0.202	0.211	30	30
		40	8	9	95.89	–	7	30	147.06	–	16	30	0.172	0.208	16	30
		64	8	9	48.890	–	30	30	60.564	–	30	30	0.140	0.205	30	30
500 (2000)	30 (6)	48	8	9	– (30)	–	0	0	269 (18)	–	3	3	1.62	1.69	21	21
		56	8	9	227 (25)	–	5	5	320 (14)	–	16	16	1.56	1.73	30	30
1000 (4000)	50 (6)	106	16	10	–	–	–	–	–	–	–	–	6.5	9.3	0	30
		122	16	10	–	–	–	–	–	–	–	–	6.5	9.2	9	30
		138	16	10	–	–	–	–	–	–	–	–	6.3	9.7	30	30
2500 (10000)	100 (6)	402	64	12	–	–	–	–	–	–	–	–	156	173	2	30
		466	64	12	–	–	–	–	–	–	–	–	102	130	20	30
		530	64	12	–	–	–	–	–	–	–	–	72.44	108	30	30
5000 (20000)	200 (6)	466	64	12	–	–	–	–	–	–	–	–	986	1164	11	30
		530	64	12	–	–	–	–	–	–	–	–	843	1102	27	30
		594	64	12	–	–	–	–	–	–	–	–	738	952	30	30

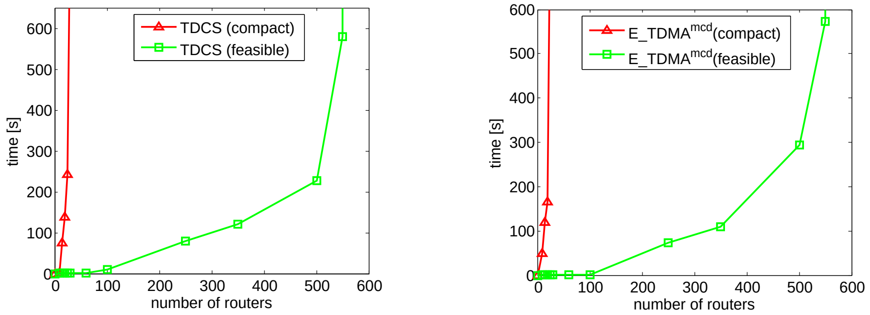
considered, and their number is shown in parentheses. Since the feasible schedule is the first solution returned by Gurobi, then the corresponding $t_{feas} < t_{com}$. The average computation time of TDCS and the E_TDMA$^{\mathrm{mcd}}$ to find the required schedule per each set of routers is demonstrated in Fig. 4.6a and Fig. 4.6b respectively. Only instances that are solved within the time limit are considered. For both exact algorithms, the average computation time exceeds the time limit for computing (i) a compact schedule for instances with more than 20 routers and (ii) a feasible schedule for instances with more than 550 routers. Furthermore, within the time limit, Gurobi failed to return (i) a compact schedule for some instances with 20 routers and (ii) a feasible schedule for some instances with at least 250 routers (more details in Sec. 4.5.3).

Regarding the heuristic algorithm, instances with larger size were solved as shown in Tab. 4.1 and in Fig. 4.6c. Also, the average computation time of the heuristic algorithm, for both compact and feasible schedules, is less than 10 s for instances up to 1000 routers, less than 3 min for instances with 2500 routers, and less than 20 min for instances with 5000 routers which proves the computational efficiency of our heuristic algorithm in comparison with both exact algorithms.

4.5.3 Success Rate of the Algorithms

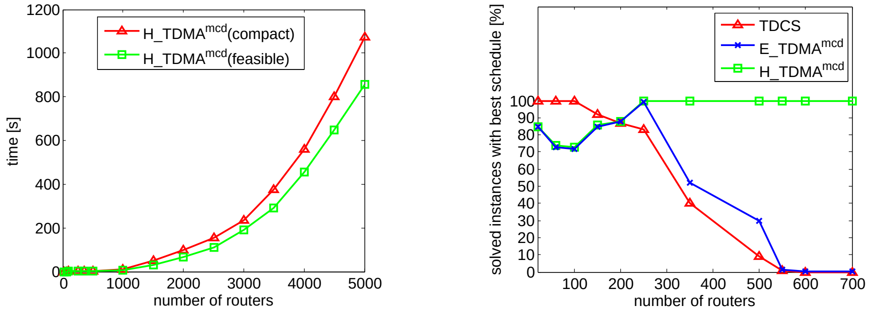
The feasible schedules returned by each algorithm are categorized into two sets. The first set, denoted by $\mathrm{F}_{\mathrm{BO}_{max}}$, includes the returned schedules with $\mathrm{BO} = \mathrm{BO}_{max}$ while the second set, denoted by $\mathrm{F}_{\leq \mathrm{BO}_{max}}$, includes all feasible schedules (i.e., the schedules with $\mathrm{BO} \leq \mathrm{BO}_{max}$). The cardinality of both sets, as given by each algorithm for each benchmark, is shown in the $\mid \mathrm{F}_{\mathrm{BO}_{max}} \mid$ and $\mid \mathrm{F}_{\leq \mathrm{BO}_{max}} \mid$ columns in Tab. 4.1 respectively. Consequently, the success rate of the algorithms is evaluated by two metrics. The first metric reveals the capability of the corresponding algorithm in returning the best known schedule while the second metric reveals the capability of the algorithm in returning a feasible schedule. By the best known schedule, we mean the feasible schedule with the maximum length of the period, given by $\mathrm{BO} \in \{\mathrm{BO}_{min}, \ldots, \mathrm{BO}_{max}\}$. Both metrics are illustrated in Fig. 4.6d and Fig. 4.6e, respectively. More specifically, The success rate of an algorithm, as shown in Fig. 4.6d, is computed as the average number of the returned best schedules per each set of routers. On the other hand, the success rate of an algorithm, as shown in Fig. 4.6e, is computed as the average number of the returned feasible schedules per each set of routers.

As shown by the three curves in Fig. 4.6d, the TDCS outperforms both E_TDMA$^{\mathrm{mcd}}$ and H_TDMA$^{\mathrm{mcd}}$ algorithms for instances up to 200 routers. This behavior is due to the fact that expressing the *e2eDeadline* as the max-



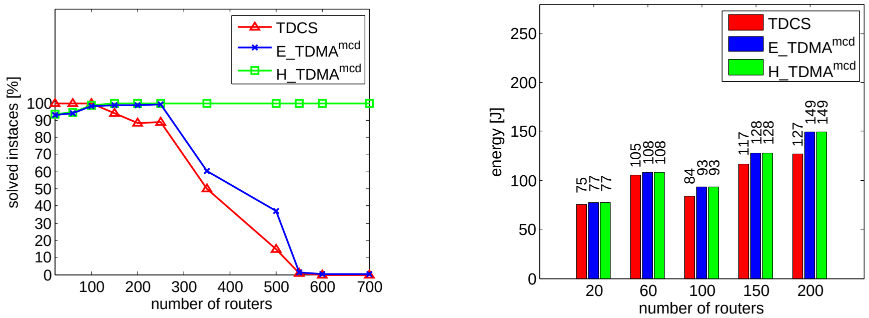
(a) TDCS computation time.

(b) E_TDMA^{mcd} computation time.



(c) H_TDMA^{mcd} computation time.

(d) Solved instances with best schedule.



(e) Solved instances.

(f) Energy consumption.

Figure 4.6: The evaluation of TDCS, E_TDMA^{mcd} and H_TDMA^{mcd} algorithms.

imum number of the crossed periods as in E_TDMA^{mcd} and H_TDMA^{mcd}, instead of seconds as in TDCS, leads to an elegant approach with shorter computation time, for the heuristic algorithm, on the cost of the inability to find a schedule with the very same value of BO as the one given by the TDCS. However, the aforementioned cost is eliminated when the *e2eDeadline* is not so tight. Both E_TDMA^{mcd} and H_TDMA^{mcd} achieve the same success rate for instances up to roughly 280 routers while for larger instances, the heuristic has the highest success rate. Moreover, both exact algorithms, TDCS and E_TDMA^{mcd}, have a success rate of 0% for instances with more than 550 routers since Gurobi fails to compute a solution within 600 s.

The curves in Fig. 4.6e indicate the improvement of the success rate of the algorithms when all solutions are considered (i.e., all feasible solutions regardless of the length of the schedule period). In such case, the TDCS achieves the highest success rate up to instances with 100 routers. Both E_TDMA$^{\text{mcd}}$ and H_TDMA$^{\text{mcd}}$ algorithms have the same success rate up to instances of size of 250 routers while for larger instances, the H_TDMA$^{\text{mcd}}$ outperforms all other algorithms.

The aforementioned results prove the capability of the heuristic to compute good solutions with longer schedule period, especially for large size instances when the exact algorithms either return a schedule with a shorter period or even fail in returning any solution within the time limit.

4.5.4 Solution Quality of the H_TDMA$^{\text{mcd}}$ Algorithm

The quality of the solution (i.e., the schedule) is evaluated by the total energy consumption of the nodes in the network. Recall that the longer the schedule period, the energy consumption of the network is reduced. The energy consumption is calculated through $U \cdot I \cdot t$ where U is the voltage, I is the current drawn and t is the execution time. The particular current drawn were given as follows: the current drawn in *receive mode* = 18.2 mA, *transmit mode* = 19.2 mA at 0 dBm, *idle mode* = 54.5 μA and *sleep mode* = 15 μA [25].

As mentioned in Sec. 4.5.3, expressing the *e2eDeadline* as the maximum number of the crossed periods in H_TDMA$^{\text{mcd}}$ might lead to a schedule with shorter period in comparison to the one given by TDCS. Consequently, the energy consumption of the network might be higher. Since TDCS has higher success rate in returning schedules with longer period up to instances with 200 routers as shown in Fig. 4.6d, we demonstrate in Fig. 4.6f the average network energy consumption within 40 min per each algorithm for those instances. The results consider only the instances that are solved by all algorithms.

As demonstrated in Fig. 4.6f, both E_TDMA$^{\text{mcd}}$ and H_TDMA$^{\text{mcd}}$ algorithms lead to the very same value of energy consumption which is compatible with the results in Fig. 4.6d. Moreover, the average extra energy consumption of the network, when E_TDMA$^{\text{mcd}}$ is used to compute the schedule, is less than 9% in comparison with H_TDMA$^{\text{mcd}}$ for instances up to 200 routers. The instances with 100 routers have less average energy consumption in comparison with instances with 60 routers due to the fact that BO$_{max}$ is set to either 4 or 8 for instances with 100 routers while it is set to either 1 or 2 for instances with 60 routers (see Tab. 4.1). For benchmarks with larger size (i.e., more than 200 routers), as illustrated in Fig. 4.6d, the H_TDMA$^{\mathrm{mcd}}$ outperforms all other algorithms in returning schedules with longer period. Hence, the corresponding schedules are more energy efficient when compared to the ones returned within the time limit by exact algorithms.

4.6 Simulation Study

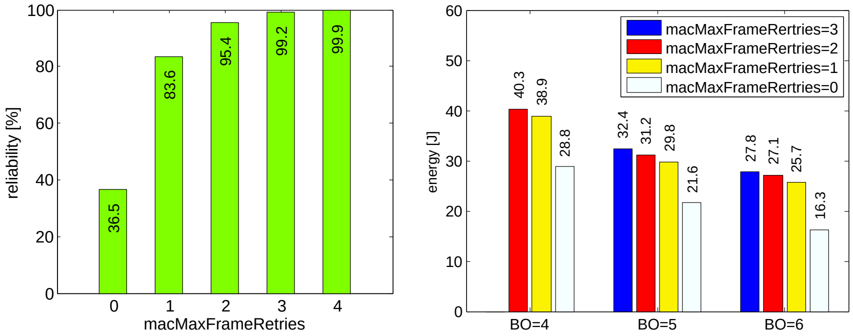
The aim of this section is to show through simulation the impact of the length of BI, as given by BO, on the energy consumption of the nodes. Moreover, due to the communication errors and the unreliability of the wireless channel, the simulation model considers both the acknowledged and unacknowledged transmissions that are supported by IEEE 802.15.4. For the acknowledged transmission the transmitter waits for a given time till it receives the acknowledgment. If waiting time elapsed while the acknowledgment is not received, the transmitter re-transmits the message. The number of re-transmissions is bounded by *macMaxFrameRetries* parameter for the proper analysis. The energy consumption of the nodes and the reliability of the transmissions as a function to the number of re-transmissions are also demonstrated. Furthermore, we illustrate the impact of the number of re-transmissions on the timeliness of the data flows within the network.

4.6.1 Simulation Scenarios

The simulation scenarios are following the example presented in Fig. 3.1 (i.e., identical to the network topology and data flows) and implemented in Opnet Modeler 17.5. The configuration parameters of each node are given by the H_TDMA$^{\mathrm{mcd}}$ algorithm. The simulation time is set to 40 min involving the generation of 2396 frames of f_1 and f_3, and 1198 frames of f_2 and f_4. Due to different kinds of disturbances, the transmission over the channel might be lost in real case WSNs. Thus, and for the sake of simplicity, we assume that the channel error rate is fixed to 20% [29]. Hence, the node drops the received message with the probability of 0.2.

4.6.2 Network Reliability

The reliability of the data transmissions within the network is calculated as the percentage of the successful transmissions from the source node to the sink and illustrated in Fig. 4.7a. Therefore, the more the reliability is, the more messages that are dispatched by the source nodes reach the sink nodes correct and on time. To guarantee a particular level of reliability for given communication error parameter over the frequency channel, the



(a) Network reliability.

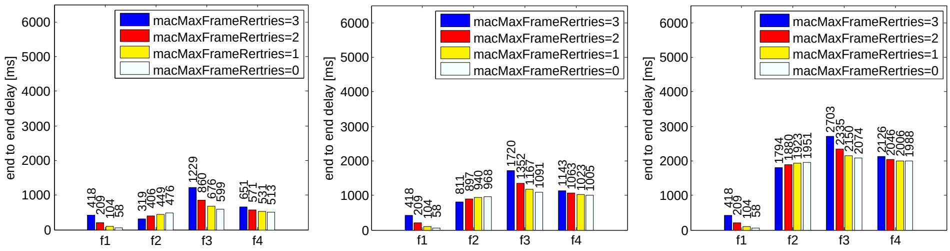
(b) Energy consumption.

Figure 4.7: The impact of the number of re-transmissions on the network reliability and the energy consumption for given length of the schedule period.

re-transmission and acknowledgment mechanism is used. Therefore, to increase the reliability of the network, the number of re-transmissions must be increased. However, the number or re-transmissions must be bounded for the proper analysis even for unknown commutation error parameter.

4.6.3 Energy Consumption

We demonstrate the energy consumption of all nodes as a function to the value of BO and the number of the re-transmissions. The energy consumption is calculated through $U \cdot I \cdot t$ where U is the voltage, I is the current drawn and t is the execution time. The particular current drawn were given as follows: the current drawn in *receive mode* = 18.2 mA, *transmit mode* = 19.2 mA at 0 dBm, *idle mode* = 54.5 μA and *sleep mode* = 15 μA [25]. In Fig. 4.7b, we depicted the case when BO $\in \{4, 5, 6\}$ and the number of re-transmissions $macMaxFrameRetries \in \{0, 1, 2, 3\}$. Notice that when BO = 4 and $macMaxFrameRetries = 3$, there is no feasible schedule. This is because increasing the number of re-transmissions increases the superframe duration of each cluster which might lead to the case at which the clusters do not fit into the schedule period. The results confirm that the energy consumption is inversely proportional to the BO value and it is directly proportional to the number of re-transmissions. Hence, a fair trade-off is required between reliability and energy efficiency. Furthermore, the results show that the longer the BI, the lower the energy consumption since the nodes spend more time in power-saving mode. The results are compatible with our objective of maximizing the length of the schedule period BI.



(a) BO = 5. (b) BO = 6. (c) BO = 7.

Figure 4.8: The impact of the number of re-transmissions and the length of the schedule period on the end-to-end delay of the data flows.

4.6.4 Timeliness of the Data Transmission

The purpose of this section is to illustrate the impact of both BO and the number of re-transmissions on the transmission end-to-end delay for each data flow in Fig. 3.1. The cluster schedule is identical to the one given in Fig. 4.1 in terms of the order of the superframe durations of the clusters. However, recall that increasing the number of re-transmissions increases the the superframe durations of the clusters and degrades the throughput within the network.

The results show that the end-to-end delay for data flows f_1, f_3 and f_4 increases when the number of re-transmissions is increasing. However, the end-to-end delay of data flow f_2, for a given value of BO, decreases by increasing the number of re-transmissions. This is because of the coloration between the schedule, as shown in Fig. 4.1, and the path of data flow f_2. In other words, the superframe of cluster C_2, the source of the data flow f_2, and the superframe of cluster C_6, the sink of the data flow f_2, are pushed towards the border of the schedule. However, the gap between both clusters is shortening when the superframe durations of all clusters within the network are prolonged. Furthermore, the results show that the end-to-end delay for all data flows increases when the length of the schedule period, as given by the value of BO, is increased. Therefore, a coloration exists between the timeliness and the energy efficiency of the network so that a fair trade-off is required.

4.7 Conclusion

In this chapter, we extended and completed the previous work presented in Chapter 3. In particular, we tackled a challenging TDMA scheduling problem which is highly appealing in control applications by considering a realistic model of multiple-collision domains Cluster-Tree topology. The traffic within the network is organized into periodic and time-constrained data flows, which may traverse the network simultaneously in opposite directions to each other. The aim is to support collision avoidance, energy efficiency, timeliness, and network reliability QoS properties. Furthermore, in contrast to Chapter 3, the spatial reuse of the transmission medium is considered to bandwidth utilization and consequently, the scalability of the network. To cope with the complexity of the ILP and to solve large size instances, we implement an efficient heuristic algorithm based on graph theory and combinatorial optimization problems. We demonstrate, through the benchmarks, that the heuristic algorithm is efficient in both computational time and solution quality.

We also implement simulation scenarios in order to emphasize the direct correlation between the reliability, energy efficiency and timeliness. As expected, the simulation results are similar to the results presented in Chapter 3. Hence, a fair trade-off between reliability, energy efficiency and timeliness is required.

Chapter 5

Optimized Distributed TDMA Scheduling Algorithms for ZigBee-Like Cluster-Tree Topology

This chapter follows the scenarios presented in Chapter 3 and Chapter 4. Therefore, it supports collision avoidance, energy efficiency, timeliness and network reliability QoS properties for large scale WSNs. However, in contrast to Chapter 3 and Chapter 4, this chapter aims to further support the on-the-fly deployment and configuration QoS property. In particular, WSNs might be deployed on the fly without the necessity of pre-existing infrastructure especially in hard to reach environments. In such a scenario, the sensor nodes are expected to self organize and configure themselves into the form of a multi-hop network so they can operate unattended. The realization of such a requirement relies mostly on the distributed methodologies that enable each node within the network to set all required parameters based on its local view of the network. However, driven by the energy efficiency, timeliness, and reliability QoS properties, this chapter relies also on the infrastructure-based WSNs such as Cluster-Tree topology. Such an assumption is realistic since the organization of the deployed nodes into Cluster-Tree can also be accomplished in a distributed manner. Therefore, this chapter considers a realistic model with single-collision domain or multiple-collision domains Cluster-Tree topology. The traffic is organized into multi-hops time-bounded data transmissions that might traverse the network with opposite direction. Consequently, this chapter proposes two distributed TDMA scheduling algorithms that solve both case studies (i.e., the case of the single-collision domain and the case of the multiple-collision domains). The algorithms allow each cluster to come up with its allocated collision-free time-slots so that the specified set of QoS properties are met. The algorithms are based on graph theory, such as distributed shortest path, distributed topological ordering, and distributed graph coloring algorithms. The efficiency of the algorithms, regarding the elapsed time to construct the schedule and energy consumption, is evaluated over benchmark instances up to several thousands of nodes. Moreover, a comparison with existing works is demonstrated in order to prove that the specified set of QoS properties are efficiently addressed in our approach.

Table 5.1: User-defined data flows parameters.

flow ID q	*source* (α_{f_q})	*sink* (β_{f_q})	*sampleSize* [bit]	*reqPeriod* [s]	*e2eDeadline* [s]	*sampleACK*
1	1	9	64	1	2	1
2	6	10	16	2	3	0
3	11	12	16	1	2	1
4	12	1	64	2	2	0

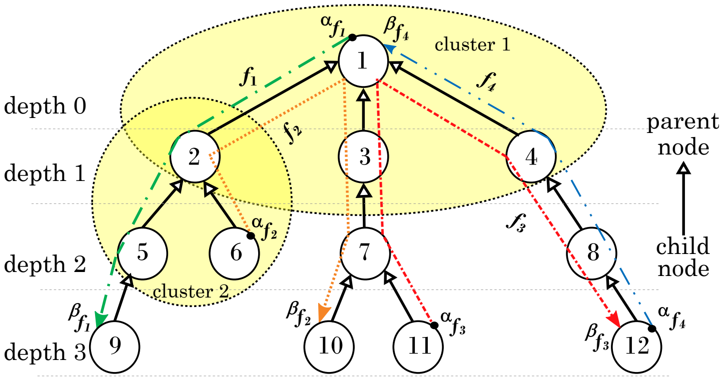


Figure 5.1: An example of Cluster-Tree WSNs with four time-constrained data flows.

5.1 Related Work

In the recent years and with the rapid utilization of WSNs in various applications, many researchers have tackled various relevant challenges that include network formation schemes, communication mechanisms, energy-efficiency, and MAC protocol configurations. Each of these issues has its own special considerations. Within the context of this chpater, we are particularly interested in works that address the centralized and distributed design of MAC protocols that are based on the TDMA approach.

Concerning the centralized TDMA approches, the scheduling problem considered in [7] is the most related to the work presented in this chapter. The authors considered Cluster-Tree topology and data flows constrained by end-to-end deadline given in time units. Then, the end-to-end deadline for each data flow is expressed as the maximum number of periods crossed by each data flow till its delivery. To allocate the time-slots to the clusters so that the timeliness requirements are met, the authors proposed a centralized exact scheduling algorithm in the case of a single-collision domain and an efficient heuristic algorithm in the case of multiple-collision domains. In this chapter, we follow the same problem statement as described in [7].

However, we propose distributed algorithms in order to enable each cluster, by itself, to come up with its allocated time-slots in the schedule. More precisely, we propose (i) a distributed counterpart to the exact centralized algorithm presented in [7] for single-collision domain, (ii) a novel centralized and heuristic scheduling algorithm for multiple-collision domains that is more suitable to distribute compared to the one presented in [7], and finally (iii) a distributed scheduling algorithm for the proposed centralized heuristic algorithm for the multiple-collision domains Cluster-Tree topology.

Furthermore, and within the context of centralized TDMA scheduling algorithms, the authors in [35] proposed a set of time-slots allocation schemes in order to improve the network throughput and to avoid the network congestion, long end-to-end communication delays and discarded messages due to buffer overflows. Simulation assessments show how the proposed allocation schemes may improve the operation of wide-scale Cluster-Tree networks. The authors in [38] proposed a TDMA scheduling algorithm for multi-hops WSNs that balances the energy consumptions among the nodes which implies the prolongation of the lifetime of the network. The authors in [42] presented a dynamic centralized TDMA scheduling algorithm for single-collision domain Cluster-Tree WSN where all transmissions, organized as data flows, are directed to the root node of the tree. The proposed algorithm enables the changing of the resource allocation of a Cluster-Tree WSN on-the-fly without imposing long inaccessibility times. Hence, the algorithm enables the network to self-adapt to changing traffic flows in order to minimize the latency. However, none of the works mentioned in this paragraph considered the case of precise timeliness requirements of data flows that might traverse the network simultaneously in the opposite direction.

Concerning the distributed TDMA scheduling approaches for single-collision domain and multiple-collision domains WSNs, several researchers assumed different scenarios with various requirements while designing distributed methods that enable the nodes in the network to allocate their time-slots within the schedule. The authors in [3] proposed an exact distributed TDMA scheduling algorithm for single-collision domain Cluster-Tree WSNs with data flows constrained by the end-to-end deadline. The algorithm is based on the centralized exact algorithm proposed in [7]. However, in this chapter, we extend and complete the previous works presented in [3] such the algorithm for a single-collision domain is improved in order to further minimize the energy consumption by the nodes during the cluster schedule construction phase. Moreover, a centralized and a distributed algorithm for the multiple-collision domains Cluster-Tree topology are proposed.

The authors in [53] considers collision-free data aggregation (i.e., all data flows are directed to single sink node) in large-scale WSNs based on the maximal independent set and graph coloring problem. The authors in

[22] designed distributed graph coloring algorithms that are utilized by the nodes in order to minimize the total number of the allocated time-slots of the nodes (i.e., the minimization of the length of the makespan of the schedule). The work presented in [11] aims at reducing the time required to acquire the schedule while supporting different modes of communication: unicast, multicast, and broadcast. The work presented in [24] proposed an energy-aware algorithm with real-time transmissions while exploiting the available flexibility in bandwidth requirements by adapting stream parameters to balance performance versus energy consumption. However, the transmission deadline is given by the maximum amount of time that can elapse between a message arrival and the completion of its transmission (i.e., it is related to intracluster communication not to the end-to-end deadline which leads to a more specific problem statement). The authors in [51] proposed both a centralized and distributed Two Way Beacon Scheduling (TWBS) algorithm in order to minimize the latency for both the upstream and downstream transmissions in ZigBee Cluster-Tree WSNs. The work assumes that all clusters allocate equal number of time-slots. Moreover, the proposed algorithms modify the original superframe structure of IEEE 802.15.4 to support two-way communications by enabling each cluster to be active twice within the schedule period. Such approach, in comparison to our proposed approach, increases the energy consumption of the nodes since each cluster-head is required to transmit two beacon frames each period rather than one frame as defined by the IEEE 802.15.4 standards.

5.2 Chapter Contributions and Outline

This chapter provides the following original contributions:

1. A realistic model with single-collision domain or multiple-collision domains Cluster-Tree topology and multi-hops data flows constrained by end-to-end deadlines expressed by the maximum number of crossed periods.
2. An exact Distributed Time Division Multiple Access for single-collision domain ($\text{DTDMA}^{\text{scd}}$) cluster scheduling algorithm. $\text{DTDMA}^{\text{scd}}$ algorithm outperforms the distributed algorithm presented in [3] in terms of energy efficiency and computation time.
3. A novel centralized heuristic algorithm that solves the time-slots allocation for the case of multiple-collision domains Cluster-Tree topology. The value of the proposed centralized algorithm, compared to the one proposed in Chapter 4, lies in its ability to be easily distributed without imposing extra calculation overheads.
4. A heuristic Distributed Time Division Multiple Access for multiple-

collision domains (DTDMA$^{\mathrm{mcd}}$) cluster scheduling algorithm which is based on our proposed centralized heuristic algorithm for the time-slots allocation sub-problem.

5. The proposed algorithms are based on the sound formulation of the problems and sub-problems concerning combinatorial optimization and graph theory. Moreover, the algorithms address the collision avoidance, energy efficiency, timeliness, network reliability, and on-the-fly deployment and configuration QoS properties.
6. The evaluation and the comparison of the proposed algorithms with the existing works over large-scale benchmarks, through various simulation scenarios, in order to demonstrate the overhead of the algorithms in term of the elapsed time to construct the schedule, the energy consumption, and network reliability.

The rest of the chapter is organized as follows. In Sec. 5.3, the system model is explained. The distributed TDMA scheduling problem is explained in Sec. 5.4. The stages of the DTDMA$^{\mathrm{scd}}$ and DTDMA$^{\mathrm{mcd}}$ are presented in Sec. 5.5, Sec. 5.6, Sec. 5.7, Sec. 5.8 and Sec. 5.9. The experimental results are presented in Sec. 5.10. Finally, we draw the conclusion in Sec. 5.11.

5.3 System Model

In this chapter, we consider the case at which the sensor nodes are deployed on-the-fly. However, we assume that the sensor nodes are self organized and configured into the form of a multi-hop Cluster-Tree WSN by using distributed clustering algorithms [54].

5.3.1 Cluster-Tree Topology

We consider a Cluster-Tree topology with n nodes and m clusters where $m < n$ (for example, the Cluster-Tree topology shown in Fig. 5.1 consists of $n = 12$ nodes and $m = 7$ clusters). The edges stand for the *parent-child* logical relations. However, the communication links between each parent and child nodes are bidirectional. Each cluster is composed of a parent node, called the cluster-head, and the set of child nodes. Consequently, each node, except the root and the leaf nodes, belongs to two clusters, once as a child node and once as a cluster-head.

Let $parent_i$ be the parent node of node i, $Child_i$ be the set of the child nodes of node i, and L be the set of the leaf nodes (i.e., $L = \{i = 1 \ldots n : Child_i = \emptyset\}$). Hence the notation C_i refers to the cluster nodes for which node i is the cluster-head (i.e., $C_i = \{i\} \cup Child_i$: $i \notin L$). In Fig. 5.1, C_1 and C_2 are illustrated by the dashed ellipses. Moreover, the notation

$SubT_i$ denotes the set of nodes rooted by node i as given in the Cluster-Tree topology. For example, $SubT_3 = \{3, 7, 10, 11\}$. In similar manner, $SubT(C_3) = \{C_3, C_7\}$. Each node i in the Cluster-Tree topology has its depth, denoted by $depth_i$, as shown in Fig. 5.1. Consequently, the depth of the cluster is given by the depth of its cluster-head node. For example, the depth of the root C_1 is 0 (i.e., $depth(C_1) = 0$).

5.3.2 Data Flow Model

Multi-hop communication within the Cluster-Tree topology are organized into data flows. Each data flow f_q may have more than one source node but exactly one sink node (see Fig. 5.1 where 4 data flows are illustrated as dashed directed lines). The source node of data flow f_q, denoted by α_{f_q}, periodically measures a sensed value with a given size and required period denoted by $sampleSize_{f_q}$ and $reqPeriod_{f_q}$ respectively and reports it to the sink node β_{f_q}. The $reqPeriod_{f_q}$ is a given parameter that is associated with data flow f_q and indicates the upper bound of the time interval between two consecutive measurements performed by the source node of the data flow.

To support applications with real time demands, each data flow is constrained by the end-to-end deadline, denoted by *e2eDeadline*, and given in time units. The *e2eDeadline* specifies the maximum allowed elapsed time between the instant when the source sends the packet to the time instant when the sink receives the packet. Moreover, to support applications with stringent reliability demands, the acknowledgment and re-transmission mechanism might be requested. The input parameter $sampleACK_{f_q}$ for data flow f_q is a boolean parameter that determines whether the acknowledgment and re-transmission mechanism is enabled or not.

Since multi-hop communication is deterministic in the Cluster-Tree topology, the packets are forwarded cluster by cluster following the unique routing path from the source cluster to the sink cluster. The source and sink clusters of f_q are determined by Def. 3.3.1 and Def. 3.3.2 and shown in Fig. 5.2

5.3.3 Cluster Life cycle

Each cluster is periodically active for a specified duration within the cluster period P. Hence, the life cycle of each cluster is divided into two portions, the active and inactive portions (see Fig. 5.3a where the life cycle of C_4 is illustrated). During the inactive portion, the cluster is in a sleep mode to save energy. The active portion of the cluster, denoted by τ, is subdivided into time-slots of equal size that are utilized by the cluster-head to exchange the data with its child nodes. The τ portion is subdivided into a CAP and

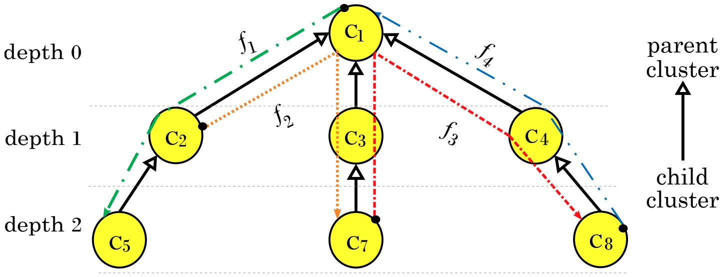


Figure 5.2: Cluster-graph that shows the source and sink clusters of each data flow in Fig. 5.1.

optional CFP. During the CAP, a slotted CSMA-CA protocol is used for the best-effort data delivery while during the CFP, the cluster-head periodically allocates time-slots to its child nodes which can be exploited for transmitting real-time traffic. The number of time-slots allocated to a given child node is relevant to the amount of data to be exchanged between the parent and that child. The time-slots used by a child node to send the data to its parent are denoted by *Tx* slots while *Rx* slots denote the time-slots for receiving the data from the parent node. P and τ are defined by two parameters as follows:

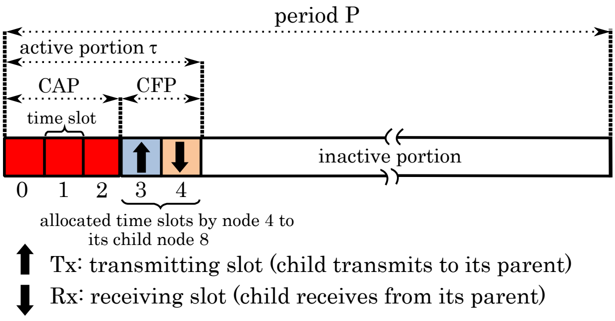
$$P= A \cdot 2^{PO}, \quad \tau= A \cdot 2^{TO} \tag{5.1}$$

where $0 \leq TO \leq PO \leq 14$ and A denotes the minimum duration of τ when $TO = 0$. For example, in the case of ZigBee Cluster-Tree topology, $A = 15.36$ ms while assuming a 2.4 GHz frequency band and 250 kbps of bit rate. We assume that all clusters have an equal P, but various τ for better bandwidth utilization.

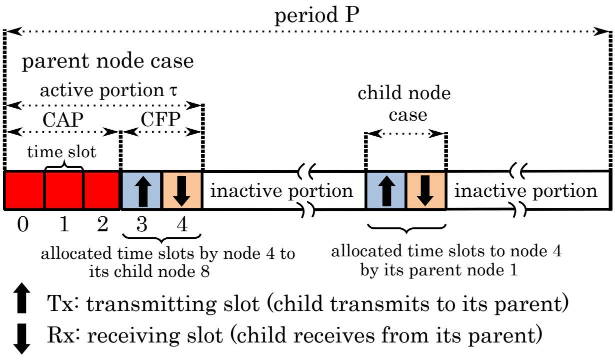
Since each node, except the root node and the leaf nodes, belongs to two clusters, then each node will be active in two cases. Once as a cluster-head and once as a child node of another cluster it belongs to. See Fig. 5.3b where the life cycle of node 4 is illustrated.

5.3.4 Multiple-Collision Domains

The spatial reuse of the transmission medium, in the case of multiple-collision domains WSNs, is crucial for better utilization of the bandwidth. The collision domain of a cluster depends on the physical deployment of the WSN nodes and the transmission and the carrier sensing areas of the cluster's nodes. Both the transmission and the carrier sensing areas depend on the strength of the radio signal and heterogeneity of the environment. The node may receive and decode the message if it is within the transmission area of the transmitter. The nodes within the carrier sensing area of


period P
active portion τ
CAP
CFP
time slot
inactive portion
0
1
2
3
4
allocated time slots by node 4 to its child node 8
Tx: transmitting slot (child transmits to its parent)
Rx: receiving slot (child receives from its parent)

(a) Life cycle of node 4 as cluster-head.


period P
parent node case
active portion τ
CAP
CFP
time slot
child node case
inactive portion
inactive portion
0
1
2
3
4
allocated time slots by node 4 to its child node 8
allocated time slots to node 4 by its parent node 1
Tx: transmitting slot (child transmits to its parent)
Rx: receiving slot (child receives from its parent)

(b) Life cylce of node 4 as cluster-head and child node.

Figure 5.3: Cluster life cylce.

the transmitter might be able to hear the transmission but cannot decode it correctly.

Let $Hear(z)$ denote the set of nodes that can hear node z and $N_f(z)$ be the set of nodes with *direct* interference with node z. Two nodes are in *direct* interference if either node can hear the other node's transmissions. Thus $N_f(z) = \{y : y \in Hear(z) \ or \ z \in Hear(y)\}$. In a similar manner, two clusters are in *direct* interference when either cluster can hear the other cluster transmissions. The interfering clusters are considered to be competitors and are not allowed to be active simultaneously. On the other hand, non-competitor clusters might be active simultaneously. For each C_u, the set $M(C_u)$, as given in Eq. 5.2 includes the competitor clusters of C_u.

$$M(C_u) = \{C_v : \exists \ z \in C_u, y \in C_v \ and \ y \in N_f(z)\} \tag{5.2}$$

5.3.5 Cyclic Nature of the Cluster Schedule

The considered scheduling problem is to determine which set of consecutive and collision-free time-slots is assigned to which cluster so that the $e2eDeadline_{f_q}$ of each f_q is met. The fact that (i) the routing problem in the tree topology is deterministic since only one path exists between any

Table 5.2: The impact of the precedence decisions on θ_{f_q}.

#	**scenario**	θ_{f_1}	θ_{f_2}	θ_{f_3}	θ_{f_4}
1	$\{(C_1,\rightarrow,C_4),(C_4,\rightarrow,C_8),(C_1,\rightarrow,C_3),(C_3,\rightarrow,C_7),(C_1,\rightarrow,C_2),(C_2,\rightarrow,C_5)\}$	0	1	2	2
2	$\{(C_1,\leftarrow,C_4),(C_4,\leftarrow,C_8),(C_1,\leftarrow,C_3),(C_3,\leftarrow,C_7),(C_1,\leftarrow,C_2),(C_2,\leftarrow,C_5)\}$	2	2	2	0
3	$\{(C_1,\rightarrow,C_4),(C_4,\leftarrow,C_8),(C_1,\leftarrow,C_3),(C_3,\leftarrow,C_7),(C_1,\leftarrow,C_2),(C_2,\rightarrow,C_5)\}$	1	2	1	1

pair of nodes and that (ii) each cluster is active only once within the schedule period as explained in Sec. 5.3.3, leads to the so-called cyclic behavior of the schedule when the multi-hops data flows have an opposite direction (more details in Chapter 3, Sec. 3.3.4). Therefore, one data flow in a cyclic cluster schedule exists, at least, that spans over multiple periods (i.e., starts in one period and ends in one of the subsequent periods). Furthermore, the end-to-end delay minimization of f_i is in contradiction with the end-to-end delay minimization of f_j when $\{f_i,\ f_j\}$ are in opposite directions.

Based on Def. 3.3.7, Def. 3.3.8, Def. 3.3.9 and Prop. 3.3.1, Tab. 5.2 illustrates the impact of the precedence decisions on the value of θ_{f_q} (i.e., the number of crossed periods for each data flow f_q) as given in Fig. 5.2.

In this chapter, we have also chosen to express the end-to-end deadline that is given in time units as the maximum number of periods to be crossed by each data flow from the time at which the transmission commences till the time at which the transmission ends. Hence, for a given f_q, the maximum number of crossed periods, denoted by $h_{f_q} \geq 0$, can be calculated as follows:

$$h_{f_q} = \left\lfloor \frac{e2eDeadline_{f_q}}{P} \right\rfloor - 1 \tag{5.3}$$

Definition 5.3.1. h_{f_q} is an integer value that associates the following constraint to each f_k: if f_q starts in the interval $[x \cdot P, (x+1) \cdot P)$, then the data has to be delivered to the sink node during or before the interval $[(x + h_{f_q}) \cdot P, (x + h_{f_q} + 1) \cdot P)$.

Definition 5.3.2. The cyclic cluster schedule satisfies the end-to-end deadline iff $\theta_{f_q} \leq h_{f_q}$ for every f_q.

Considering $P = 1$ s and the *e2eDeadline* as given in Tab. 5.1, we get: $h_{f_1} = 1$, $h_{f_2} = 2$, $h_{f_3} = 1$ and $h_{f_4} = 1$.

5.4 Distributed TDMA Scheduling Algorithm

The key problem to solve in this chapter is to allocate, in a distributed manner, a set of consecutive and collision-free time-slots to each cluster so that $\theta_{f_q} \leq h_{f_q}$ for each f_q. To support the applications with stringent energy-efficacy demands, our objective is to maximize the life time of the network by maximizing the length of the inactive portion of each cluster. Since the active portion of each cluster is given by the payload of the data to be exchanged within the cluster, the inactive portion is prolonged when the length of the schedule period, denoted by P, is prolonged. Thus, our objective is equivalent to the maximization of P. However, since h_{f_q} is inversely proportional to P (see Eq. (5.3)), then the longer the P, the harder is to satisfy the h_{f_q} for each f_q. Hence, P has to be set to the largest possible so that h_{f_q} for each f_q is met.

The complexity of the underling scheduling problem while considering the case of a single-collision domain Cluster-Tree topology is polynomial while it turns into $\mathcal{NP}$-hard problem when multiple-collision domains Cluster-Tree topology is considered [7]. Thus, in this chapter, we propose two distributed TDMA scheduling algorithms. The exact Distributed Time Division Multiple Access for a single-collision domain (DTDMA $^{\text{scd}}$) Cluster-Tree topology and a heuristic Distributed Time Division Multiple Access for multiple-collision domains (DTDMA$^{\text{mcd}}$) Cluster-Tree topology. Both scheduling algorithms consist of 5 stages. Stages $1, 2, 3$ and 5, explained in Sec. 5.5, Sec. 5.6, Sec. 5.7 and Sec. 5.9 respectively, are identical for both algorithms while stage 4 is specific for each algorithm as explained in Sec. 5.8. For each stage, we present the basic principles followed by the distributed algorithm. We assume that the network topology has been set up and that each node i maintains the following input parameters:

1. id: a unique identifier assigned to the node; $id = i$.
2. $parent_i$: the parent node of the node i.
3. $Child_i$: the set of child nodes of node i.
4. $depth_i$: the depth of the node i in the Cluster-Tree topology.
5. t_s: the duration of the fixed-size time-slot within τ_i.
6. $\#TS_{csma}$: the number of the time-slots within the CAP portion of τ_i.

5.5 The Calculation of P_{max} and the Number of Clusters m

At this stage, both the upper bound of the length of the schedule period P_{max} and the number of clusters m are calculated. P_{max} and m are considered as input parameters within the second stage as will be explained in Sec. 5.6.2. The value of P_{max} is given by Eq. (5.1) by substituting PO by PO_{max}. The value of PO_{max} is bounded by the shortest $reqPeriod_{f_q}$ among all of the data

Algorithm 6: The distributed calculation of the number of clusters and the upper bound of the schedule period.

```
1  $m_i \leftarrow 0,\ ch_i \leftarrow 0$
2  $Y_i \leftarrow \{f_q : \alpha_{f_q} = i\}$
3  if $Y_i = \emptyset$ then
4      $PO^i_{max} \leftarrow 14$
5  else
6      $reqP_i \leftarrow \min_{f_q \in Y_i}(reqPeriod_{f_q})$
7      $PO^i_{max} \leftarrow \min\left(14, \left\lfloor \log_2\left(\frac{reqP_i}{A}\right)\right\rfloor\right)$
8  $pck \leftarrow$ receive packet from node $j \in Child_i \cup \{parent_i\}$
9  switch $pck.type$ do
10     case $RP\text{-}NC$ do
11         $PO^i_{max} \leftarrow \min(PO^i_{max}, pck.period\_order)$
12         $m_i \leftarrow m_i + pck.num\_clusters$
13         $ch_i \leftarrow ch_i + 1$
14         if $ch_i = |Child_i|$ then
15             if $i \neq 1$ then
16                 $pck \leftarrow \text{RP-NC}(i, parent_i, m_i + 1, PO^i_{max})$
17                 send pck to my parent node
18             else
19                 $m_i \leftarrow m_i + 1$
20                 foreach $k \in Child_i$ do
21                     $pck \leftarrow \text{P-NC}(i, k, m_i, PO^i_{max})$
22                     send pck to node k
23     case $P\text{-}NC$ do
24         $PO^i_{max} \leftarrow pck.period\_order$
25         $m_i \leftarrow pck.num\_clusters$
26         foreach $k \in Child_i$ do
27             $pck \leftarrow \text{P-NC}(i, k, m_i, PO^i_{max})$
28             send P-NC to node k
```

flows as shown in Eq. (5.4).

$$PO_{max} = \min\left(14, \left\lfloor \log_2\left(\frac{\min_q(reqPeriod_{f_q})}{A}\right)\right\rfloor\right) \tag{5.4}$$

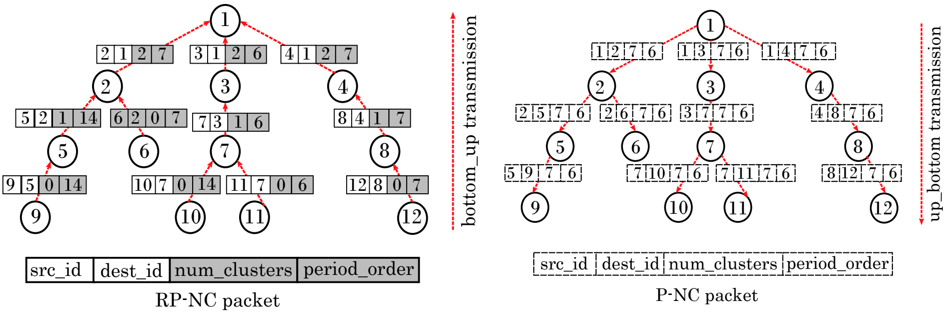
Simply, the value of m is given by Eq. (5.5) where n is the number of nodes in the given Cluster-Tree topology while L is the set of leaf nodes.

$$m = n - \mid L \mid \tag{5.5}$$

Alg. 6 demonstrates the mechanism used by each node in the network to determine the value of PO_{max} and m in a distributed manner. At lines 1 to 7, the initialization of PO_{max} and m, by each node, is shown. Then two phases of packet transmissions are commenced. The first phase is triggered as the bottom-up pattern so that each node receives one packet, denoted by the Required Period-Number of Clusters (RP-NC), from each child node and then sends RP-NC packet to its parent node as shown at lines 10 to 22. The RP-NC packet consists of the following fields:

Table 5.3: The initial value of PO_{max} and m for the example in Fig. 5.1.

nodes	PO_{max}	m
$\{2, 3, 4, 5, 7, 8, 9, 10\}$	14	0
$\{1, 11\}$	6	0
$\{6, 12\}$	7	0



(a) The transmission of RP-NC packets . (b) The transmission of P-NC packets

Figure 5.4: Number of clusters and period calculation.

1. *src_id*: the *id* of the sender.
2. *dest_id*: the *id* of the receiver.
3. *num_clusters*: this field is set to the number of clusters that belong to the subtree rooted by the sender based on the received RP-NC packets from the child nodes.
4. *period_order*: this field is set to the minimum value of PO_{max} among the nodes that belong to the subtree rooted by the sender based on the received RP-NC packets from the child nodes.

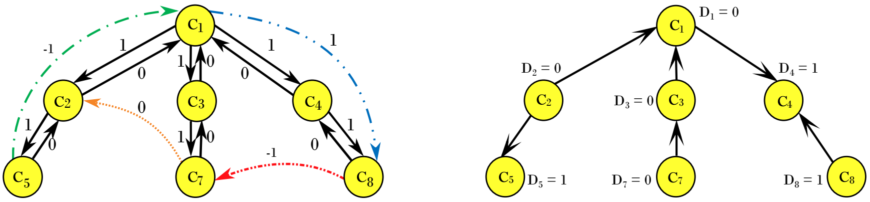
The *num_clusters* and *period_order* fields are set as shown at lines 11 to 12. Hence, Once a node i receives a RP-NC from all its child nodes, it realizes the number of clusters and the minimum value of PO_{max} within the the subtree rooted by node i. Consequently, when node 1 receives RP-NC from all its child nodes, the desired value of m and PO_{max} are computed and the first phase terminates.

The second phase, as shown at lines 23 to 28, is triggered as the top-bottom pattern. Each node receives one packet, denoted by the Period-Number of Clusters (P-NC), from its parent node and then sends P-NC to each child node. The P-NC packet composes the same fields as RP-NC where the *num_clusters* and *period_order* fields are set to the m and PO_{max} values as computed by node 1 during the first phase, respectively.

Considering the example presented in Fig. 5.1, the *reqPeriod* parameter

$0 \leq D_j - D_i \leq 1$	topological constraints
$D_1 - D_5 \leq -1$	for data flow f_1
$D_2 - D_7 \leq \ \ 0$	for data flow f_2
$D_7 - D_8 \leq -1$	for data flow f_3
$D_8 - D_1 \leq \ \ 1$	for data flow f_4

Figure 5.5: The constraint model for the example in Fig. 5.1.



(a) Inequality graph.

(b) POCA graph.

Figure 5.6: Determination of the precedence decisions.

as shown in Tab. 5.1 and setting $A = 15.36$ ms, then Tab. 5.3 depicts the initial values of PO_{max} and m as set by each node, while Fig. 5.4a and Fig. 5.4b illustrate the first and the second phase of this stage respectively.

5.6 Determining the Precedence Decisions

As explained in Prop. 3.3.1 and Def. 5.3.2, the precedence decisions between every two consecutive clusters on the path of f_q, as will be realized by the cluster schedule, is the key problem to be solved so that $\theta_{f_q} \leq h_{f_q}$ of each f_q.

To solve the precedence decision determination problem, we use the constraint model presented in Chapter 3 (Fig. 3.8). The constraint model for the example shown in Fig. 5.2 is presented in Fig. 5.5. To calculate D_i, we presented in Sec. 3.4.3 an exact polynomial centralized algorithm which consists of two parts:

1. construction of the inequality graph $Q(V, E)$ where V is the set of clusters while for each constraint $D_j - D_i \leq const$, an edge is added from cluster C_i to cluster C_j and weighted by a *const*. Hence, each edge $e(C_i, C_j) \in E(Q)$, weighted by $c_{i,j} \in \mathbb{Z}$, represents the constraint $D_j - D_i \leq c_{i,j}$. The inequality graph, for the constraints in Fig. 5.5, is depicted in Fig. 5.6a where the dashed edges, denoted as $E^f(Q) \in E(Q)$, represent the data flow constraints, while the solid edges, denoted by $E^g(Q) \in E(Q)$, represent the topological constraints. Hence, $E(Q) = E^f(Q) \cup E^g(Q)$.
2. The value of D_i, for each cluster C_i, equals the length of the shortest path from C_1 to C_i in Fig. 5.6a.

Since some edges in Fig. 5.6a may have negative weights, a negative cycle may exist and consequently, the constraint model is infeasible. Therefore, the nonexistence of a negative cycle is a necessary condition for the feasibility of the precedence determination problem. Solving the shortest path problem in Fig. 5.6a, we get $D = (0, 0, 0, 1, 1, 0, 1)$ and the resulting chosen precedence decisions are identical to the one depicted by the POCA graph shown in Fig. 5.6b.

In Sec. 5.6.1 and Sec. 5.6.2, we present the distributed algorithms that enable each cluster to determine the precedence decision between itself and its parent and child clusters so that the *e2eDeadline* of each data flow is met. Essentially, two algorithms are proposed: the first one deals with the construction of the inequality graph while the second one deals with the distributed calculations of D_i for each C_i.

5.6.1 Construction of the Inequality Graph

The distributed construction of the inequality graph, and later on solving the shortest path problem, by observing Fig. 5.6a, requires that each C_i, namely the cluster-head, computes the following sets:

1. F_i^l: the set that includes the *ids* of the clusters that are the heads of the dashed edges leaving C_i in the inequality graph. Hence, $F_i^l = \{v : e(C_i, C_v) \in E^f(Q)\}$. For example, in Fig. 5.6a, $F_1^l = \{8\}$ and consequently $F^l(C_1) = \{C_8\}$.
2. F_i^e: the set of tuples where the first entry of each tuple includes the *ids* of the clusters that are the tails of the dashed edges entering C_i in the inequality graph. The second entry is the weight associated with that dashed edge. Hence $F_i^e = \{(v, c_{v,i}) : e(C_v, C_i) \in E^f(Q)\}$. In Fig. 5.6a, $F_1^e = \{(5, -1)\}$ and consequently $F^e(C_1) = \{(C_5, -1)\}$.
3. G_i^e: the set of tuples where the first entry of each tuple is the *id* of the cluster which is the tail of one solid edge that is entering C_i in the inequality graph. The second entry is the weight associated with that solid edge. Hence, $G_i^e = \{(j, c_{j,i}) : e(C_j, C_i) \in E^g(Q)\}$. Notice that $c_{j,i} = 0$ if $C_i = \mathit{parent}(C_j)$, $c_{j,i} = 1$ if $C_i \in \mathit{Child}(C_j)$. In Fig. 5.6a, $G_2^e = \{(1, 1), (5, 0)\}$. The cost is maintained by the cluster-head of C_i so that $c_{j,i} = 0$ if $C_i = \mathit{parent}(C_j)$, $c_{j,i} = 1$ if $C_i \in \mathit{Child}(C_j)$.

Since each node maintains its parent and child nodes, then G_i^e is simply constructed as explained previously. To accomplish the F_i^l and F_i^e calculations, four packets are utilized: FLOW-INFO, FLOW-ACK, READY and STOP packets. The FLOW-INFO packet is transmitted by each source node of a data flow to the sink node while the FLOW-ACK packet is transmitted by the sink node to the source node once the FLOW-INFO packet is received by the sink node. The READY and STOP packets are used for synchronization purposes. The pseudo-code demonstrating the transmission of the aforementioned packets is shown in Alg. 7. The FLOW-INFO packet, transmitted by node α_{f_q} to node β_{f_q}, consists of the following fields:

1. *src_id*: it is set to the *id* of the node α_{f_q}.
2. *dest_id*: it is set to the *id* of the node β_{f_q}.
3. *cluster_src_id*: it is set to the *id* of the source cluster of f_q, denoted by $C_{src_{f_q}}$, as given by Def. 3.3.1.
4. *num_down_hops*: this field is used to determine the number of downstream hops on the path of f_q from $C_{src_{f_q}}$ to $C_{sink_{f_q}}$. Hence, it is initialized to 1 if $src\_id = cluster\_src\_id$ (i.e., $\alpha_{f_q} = src_{f_q}$); otherwise to 0.
5. *deadline*: this field is set to the value of the $e2eDeadline_{f_q}$ as given in time units. This field is required just in case $src\_id \neq cluster\_src\_id$; otherwise it is omitted.

The FLOW-INFO packet traverses the unique path from node α_{f_q} to node β_{f_q}. Whenever a *parent-child* hop is encountered on the path of the FLOW-INFO from node src_{f_q} to node $sink_{f_q}$, the *num_down_hops* field is incremented by 1 as shown in Alg. 7 line 25. Once the FLOW-INFO packet reaches the sink cluster, the node $sink_{f_q}$ includes the node src_{f_q} into the $F^l_{sink_{f_q}}$ set as shown at lines 19 to 21.

Once the node β_{f_q} receives the FLOW-INFO packet, then as shown at lines 17 to 18, it sends FLOW-ACK packet to node α_{f_q} that consists of the following fields:

1. *src_id*: it is set to the *id* of the node β_{f_q},.
2. *dest_id*: it is set to the *id* of the node α_{f_q}.
3. *cluster_sink_id*: it is set to the *id* of the sink cluster of f_q, denoted by $sink_{f_q}$, as given by Def. 3.3.2.
4. *num_down_hops*: it is set to the *num_down_hops* field of the FLOW-INFO packet as received by node β_{f_q}.

The FLOW-ACK packet traverses the unique path from node β_{f_q} to node α_{f_q}. Once node src_{f_q} receives the FLOW-ACK, then as shown at lines 28 to 36, it firstly calculates the value of c_{f_q} as given by Eq. (3.12) and then it checks whether the tuple $(sink_{f_q}, -)$ is already included into $F^e_{src_{f_q}}$. If so, $F^e_{src_{f_q}}$ is updated so that the tuple with minimum value of c_{f_q} is kept. This is consistent with our aim of solving the shortest path problem within the consecutive phase as explained in Sec. 5.6.2. Otherwise, the tuple $(sink_{f_q}, c_{f_q})$ is added to the $F^e_{src_{f_q}}$ set.

As mentioned previously, the READY and STOP packets are used for synchronization purposes. The READY packet is transmitted by node i

to its parent node $j = parent_i$ when it receives READY packet from each node $k \in Child_i$ and FLOW-ACK packet from each node $\beta_{f_q} : \alpha_{f_q} = i$ (see Alg. 7 at line 43 and lines 50 to 51). Thus the transmission of the READY packets is triggered from the leaf nodes towards the root node of the tree. Moreover, by transmitting the READY packet, the node indicates its readiness to execute the next stage of the algorithm. Once the root node, node 1, receives READY packets from its child nodes and the FLOW-ACK packets if any are required (i.e., if node 1 is a source node of a data flow), node 1 sends STOP packet to its child nodes (see Alg. 7 at lines 48 to 48). Once a node receives the STOP packet, it sends STOP to each child node and terminates the construction of the inequality graph and (see Alg. 7 lines 45 to 46). Fig. 5.7 illustrates the transmission of the FLOW-INFO and FLOW-ACK packets in a Cluster-Tree topology with 6 nodes and 2 data flows.

In comparison with [3], the length of the FLOW-INFO and the length of FLOW-ACK packets are reduced by omitting the fields that store the depth of the parent node of both the source and sink nodes, the data flow type (i.e., upstream, downstream, or bidirectional), and the depth of the node at which the data flow changes its direction in the case of bidirectional data flows. Such a reduction was the direct result of the proposed modification on the ILP model compared by the ILP model used in [3]. Moreover, the reduction significantly simplifies our proposed algorithm on one hand and on the other hand, it improves the performance of our proposed algorithm in terms of energy consumption since the shorter the packet, the shorter the transmission time and consequently the energy consumption decreases.

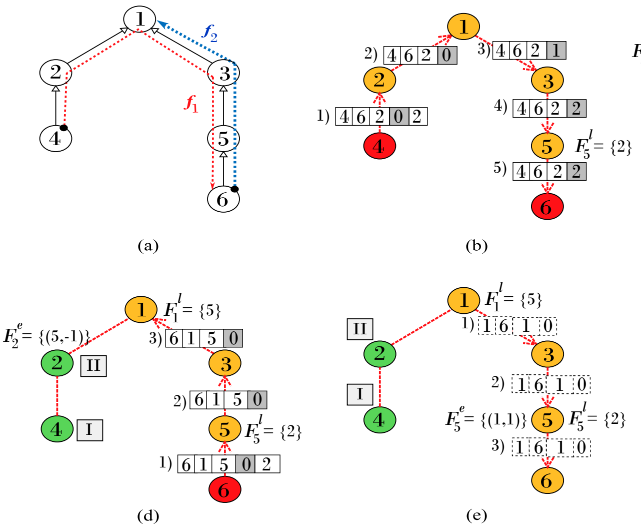
5.6.2 The Calculation of D_i

The D_i value of each C_i is equal to the length of the shortest path from C_1 to C_i in the inequality graph as explained previously in this section (see Fig. 5.6a as an example). Thus during this stage, only the cluster-head nodes are involved (i.e., the leaf nodes are not part of the calculations as shown in Fig. 5.6a). In fact, each leaf node is in a waiting state until it receives its allocated time-slots by its parent node that takes place at stage 5 as will be explained in Sec. 5.9.

The Distance Vector (DV) algorithm [20], a distributed version of the Bellman-Ford algorithm, is one approach for the distributed calculation of the shortest path tree in a given graph [12]. However, the DV has the following limitations concerning the specification of the shortest path tree problem in the inequality graph: (i) It is an asynchronous algorithm, and since our proposed distributed TDMA scheduling algorithm consists of multiple stages, then synchronization between stages is required. (ii) It assumes

Algorithm 7: The construction of the inequality graph

```
$ch_i \leftarrow 0$, $Y_i \leftarrow \{f_q : \alpha_{f_q} = i\}$, $y_i \leftarrow 0$, $F_i^l = \emptyset$, $F_i^e = \emptyset$, $quit_i \leftarrow false$
$Src_i = \emptyset$ // set of triples $(\alpha_{f_q}, \beta_{f_q}, e2eDeadline_{f_q}) : i = \alpha_{f_q} = src_{f_q}$
foreach $f_q \in Y_i$ do
    $src_{f_q} = getSrcClusterId()$
    if $i = src_{f_q}$ then
        $pck \leftarrow FLOW\text{-}INFO(i, \beta_{f_q}, 1, src_{f_q})$
        $Src_i \leftarrow Src_i \cup \{(i, \beta_{f_q}, e2eDeadline_{f_q})\}$
    else
        $pck \leftarrow FLOW\text{-}INFO(i, \beta_{f_q}, 0, src_{f_q}, e2eDeadline_{f_q})$
    send pck to node $\beta_{f_q}$
while $(\neg quit_i)$ do
    $pck \leftarrow$ receive packet from node $j : C_j \in Child(C_i) \cup \{parent(C_i)\}$
    switch $pck.type$ do
        case FLOW-INFO do
            if $i = pck.cluster\_src\_id$ then
                $Src_i \leftarrow Src_i \cup \{(pck.src\_id, pck.dest\_id, pck.deadline)\})$
            if $i = pck.dest\_id$ then
                send FLOW-ACK packet to the node $pck.src\_id$
            else if $pck.dest\_id \in Child_i$ then
                add node $pck.cluster\_src\_id$ to $F_i^l$
                send pck to node $pck.dest\_id$
            else
                $k \leftarrow nextHop()$
                if $k \in Child_i$ then
                    $pck.num\_down\_hops \rightarrow pck.num\_down\_hops + 1$
                send pck to node $k$
        case FLOW-ACK do
            $a \leftarrow (pck.dest\_id, pck.src\_id, -)$
            if $a \in Src_i$ then
                $e2eDeadline_f = a_3$// get the third entry of the triple
                $h_f = \left\lfloor \frac{e2eDeadline_f}{P} \right\rfloor - 1$
                $c_f \leftarrow h_f - pck.num\_down\_hops$
                if $(pck.cluster\_sink\_id, -) \in F_i^e$ then
                    $update(F_i^e)$
                else
                    add $(pck.cluster\_sink\_id, c_f)$ to $F_i^e$
            if $i = pck.dest\_id$ then
                $y_i \leftarrow y_i + 1$ // increase number of received FLOW-ACK packets
            else
                $k \leftarrow nextHop()$, send pck to node $k$
            break
        case READY do
            $ch_i \leftarrow ch_i + 1$
            break
        case STOP do
            send STOP to child nodes, $quit_i \leftarrow true$
    if $ch_i = \mid Child(C_i) \mid$ and $y_i = \mid Y_i \mid$ then
        if $i = 1$ then
            send STOP packet to child nodes, $quit_i \leftarrow true$
        else
            send READY packet to the $parent_i$ node, $ch_i \leftarrow -1$
```



flow_id *q*	*e2eDeadline* [s]	*P* [s]
1	2	1
2	2	1

(g)

src_id	dest_id	cluster_src_id	num_down_hops	deadline

(h)

src_id	dest_id	cluster_sink_id	num_down_hops

(i)

🔴 source node of data flow (i.e., $\Upsilon_i \neq \Phi$) 🟡 not source node of a data flow (i.e., $\Upsilon_i = \Phi$) 🟢 ($x_i = |child_i|$ *and* $y_i = |\Upsilon_i|$)

Figure 5.7: The distributed construction of the inequality graph. (a): a Cluster-Tree topology with 6 nodes and two data flows (b): transmission of FLOW-INFO packet from node 4 to node 6, (c): transmission of FLOW-ACK from node 6 to node 4, (d): transmission of FLOW-INFO from node 6 to node 1, (e): transmission of FLOW-ACK from node 1 to node 6, (f): the order of the nodes in sending the READY packets, (g): the user defined data flows parameters for the example in Fig. 5.7a. (h): FLOW-INFO packet format, (i): FLOW-ACK packet format.

that each edge in the graph is a communication link between the involved nodes which is not compatible with the underling inequality graph since the dashed edges in the inequality graph are not communication links. (iii) The existence of negative cycles is not detected.

To cope with DV limitations, we propose a modified version of the DV as shown in the pseudo-code in Alg. 8. Recall that Alg. 8 is executed only by each cluster-head, node $i \notin L$, in the network. Consequently, each node initializes PO to PO_{max}, as calculated in Sec. 5.5, and the algorithm iterates till a feasible solution is found or PO becomes less than 0. The *feasible* variable denotes whether this stage terminates with a feasible solution or a negative cycle has been detected. Each node i initializes D_i to $depth_i$. Moreover, to determine the precedence decision between each cluster and its child and parent clusters, H_i is a data structure used by each node $i \notin L$ to store the value of $D_j : C_j \in \{parent(C_i)\} \cup Child(C_i)$. The initialization of H_i is shown in Alg. 8 at lines 4 to 4. Then the algorithm, for a given PO, iterates for, at most, $m+1$ iterations ($m-1$ iterations, at most, for the calculation of D_i, one iteration for detecting the existence of negative cycles and one additional iteration at which each parent node informs its child nodes whether a feasible solution has been found).

The D_i calculation mainly utilizes two types of packets. The Single Hop Distance Value (SHDV) and the Multiple Hops Distance Value (MHDV). Both packets have the same format that includes the *id* of the sender *src_id*, the *id* of the receiver *dest_id* and the estimated shortest distance of the sender denoted by *est_dist*. Once a node i receives SHDV or MHDV packet, for simplicity denoted by *pck*, from node j so that $i = pck.dest\_id$, Eq. (5.6) is applied:

$$D_i = min\{D_i, pck.est\_dist + c_{j,i}\} : pck.est\_dist = D_j \tag{5.6}$$

Morover, the H_i data structure is updated whenever node i receives SHDV packet from its parent or child nodes. To accomplish the synchronization purposes at each iteration, the transmissions of SHDV and MHDV by node $i \notin L$, are commenced in the following order as shown in Alg. 8 lines 18 to 22:

1. node i receives SHDV from its parent node j.
2. node i sends SHDV to its parent node j and to each child node $k : C_k \in Child(C_i)$.
3. node i sends MHDV to each node $u \in F_i^l$.

Notice that sending the SHDV, at each iteration, is firstly commenced by node 1 since $parent_1 = \emptyset$. The node, that sends and receives all required SHDV and MHDV packets, sends READY packet to its parent node when it receives READY packets from all its child nodes. Hence, each iteration, at each node, lasts until the node sends READY packet to its parent node.

If an iteration terminates with no further changes of D_i, the D_i calculation can be terminated, as subsequent iterations will not lead to further changes. The termination in such a case avoids sending and receiving the redundant SHDV and MHDV packets. Moreover, the D_i calculation must be terminated in case a negative cycle has been detected. Alg. 8, precisely lines 30 and 32 in addition to lines 40 to 49, deals with both aforementioned cases of termination. Consequently, the READY packet, utilized at this stage, is extended to include, in addition to the *src_id* and *dest_id* fields, the following two fields:

Algorithm 8: The distributed calculation of D_i.

```
$PO_i \leftarrow PO_{max}$
while $PO_i \geq 0$ do
    $feasible_i \leftarrow false,\ D_i^0 \leftarrow depth_i,\ t_i \leftarrow 1,\ H_i.nodes \leftarrow \{j : j \in Child_i \cup parent_i\}$
    $H_i.D_j \leftarrow D_i + 1 : j \in Child_i, \quad H_i.D_j \leftarrow D_i - 1 : j = parent_i$
    while $t_i \leq m_i + 1$ do
        $quit_i \leftarrow false,\ D_i^{t_i} \leftarrow D_i^{t_i - 1},\ noChange_i \leftarrow true,\ negCycle_i \leftarrow false$
        if $i = 1$ then
            if $t_i \leq m_i$ then
                send SHDV pck to the child nodes
            else
                send STOP pck to child nodes, $quit_i \leftarrow true$
        while $\neg quit_i$ do
            $pck \leftarrow$ receive packet form node $j \in Child_i \cup \{parent_i\}$
            switch $pck.type$ do
                case SHDV do
                    $D_i^{t_i} \leftarrow min\{D_i^{t_i}, pck.est\_dist + c_{v,i}\} : v = pck.src\_id,$
                    $update(H_i)$
                    if $j = parent_i$ then
                        send SHDV to my parent and my child nodes
                    else if SHDV is received from every child then
                        send MHDV to each node $u \in F_i^l$
                    break
                case MHDV do
                    if $i = pck.dest\_id$ then
                        $D_i^{t_i} \leftarrow min\{D_i^{t_i}, pck.est\_dist + c_{v,i}\} : v = pck.src\_id$
                    else
                        $k \leftarrow nextHope(),$ Route pck to node $k$
                    break
                case READY do
                    $noChange_i \leftarrow (noChange_i$ and $pck.no\_change)$
                    $negCycle_i \leftarrow (negCycle_i$ or $pck.neg\_cycle)$
                    break
                case STOP do
                    if $pck.feasible = false$ then
                        $PO_i \leftarrow PO_i - 1,$ update $c_{v,i}$ for each node $v \mid (v, \_) \in F_i^e$
                    else
                        $feasible_i \leftarrow true,$ call Alg. 9
                    send STOP to every child node, $t_i \leftarrow m_i + 2, \quad quit_i \leftarrow true$
                    break
            if toQuit() then
                $noChange_i \leftarrow (noChange_i$ and $(D_i^{t_i} = D_i^{t_i - 1}))$
                $negCycle_i \leftarrow (negCylce$ or $(D_i^{t_i} < 0))$
                if $i = 1$ then
                    $feasible_i \leftarrow (noChange_i$ and $\neg\ negCycle_i)$
                    if $(noChange_i = true$ or $negCycle_i = true)$ then
                        send STOP packet to child nodes, $t_i \leftarrow m_i + 2$
                else
                    send READY to parent node
                $quit_i \leftarrow true$
        $t_i \leftarrow t_i + 1$
    if $feasible_i = true$ then
        break
```

Algorithm 9: The distributed determination of the successor and the predecessor clusters.

```
$Succ_i \leftarrow \emptyset$, $Pred_i \leftarrow \emptyset$
foreach $j \in H_i.nodes$ do
    if $j = parent_i$ then
        if $D_i = H_i.D_j$ then
            $Succ_i \leftarrow Succ_i \cup \{j\}$
        else
            $Pred_i \leftarrow Pred_i \cup \{j\}$
    else
        // $j \in$ Child$_i$
        if $D_i = H_i.D_j$ then
            $Pred_i \leftarrow Pred_i \cup \{j\}$
        else
            $Succ_i \leftarrow Succ_i \cup \{j\}$
```

1. *no_change*: this field is set to 1 by node i if no changes in the value of D_k have been encountered at any node $k : C_k \in subT(C_i)$; otherwise to 0.
2. *neg_cylce*: this field is set to 1 by node i if a negative cycle is detected by any node $k : C_k \in subT(C_i)$; otherwise to 0.

Hence, the READY packet, sent by a node i to its parent j at the t^{th} iteration, also indicates whether at least one node $k : C_k \in subT(C_i)$ exists which has changed its D_k value (i.e., $no\_chagne = 0$ *iff* $\exists k : C_k \in subT(C_i), D_k^t \neq D_k^{t-1}$). Otherwise, the field *no*_change is set to 1. Similarly, if a negative cycle has been detected by node i then node i informs its parent node through the READY packet by setting the field *neg_cycle* to 1. Node i detects the existence of a negative cycle in the following cases:

1. D_i becomes negative at any iteration. i.e., $D_i^t < 0 : t < m$.
2. node i updates its D_i at the m^{th} iteration. i.e., $D_i^{t-1} \neq D_i^t : t = m$.

Thus, when node 1 receives the required READY packets from its child nodes, it is fully aware whether to terminate the D_i calculation stage or not. If termination is the case, then node 1 sends STOP packet that traverses the network. The STOP packet contains, in addition to the *src_id* and *dest_id* fields, the *feasible* field that denotes whether a feasible solution has been found (i.e., no negative cycles are encountered). Once a node i receives the STOP packet from its parent node, then as shown in Alg. 8 lines 33 to 39, it checks the *feasible* field. If $feasible = 1$, the node calls Alg. 9 to determine the precedence decisions between C_i and its parent and child clusters based on the H_i. Hence, the $Succ_i$ and $Pred_i$ sets are dedicated to include the successor and predecessor cluster-head nodes to the node i, respectively. On the other hand, when $feasible = 0$, then node i decreases the value of PO_i by 1 and consequently it updates the value of $c_{v,i} : \forall(v, -) \in F_i^e$ and

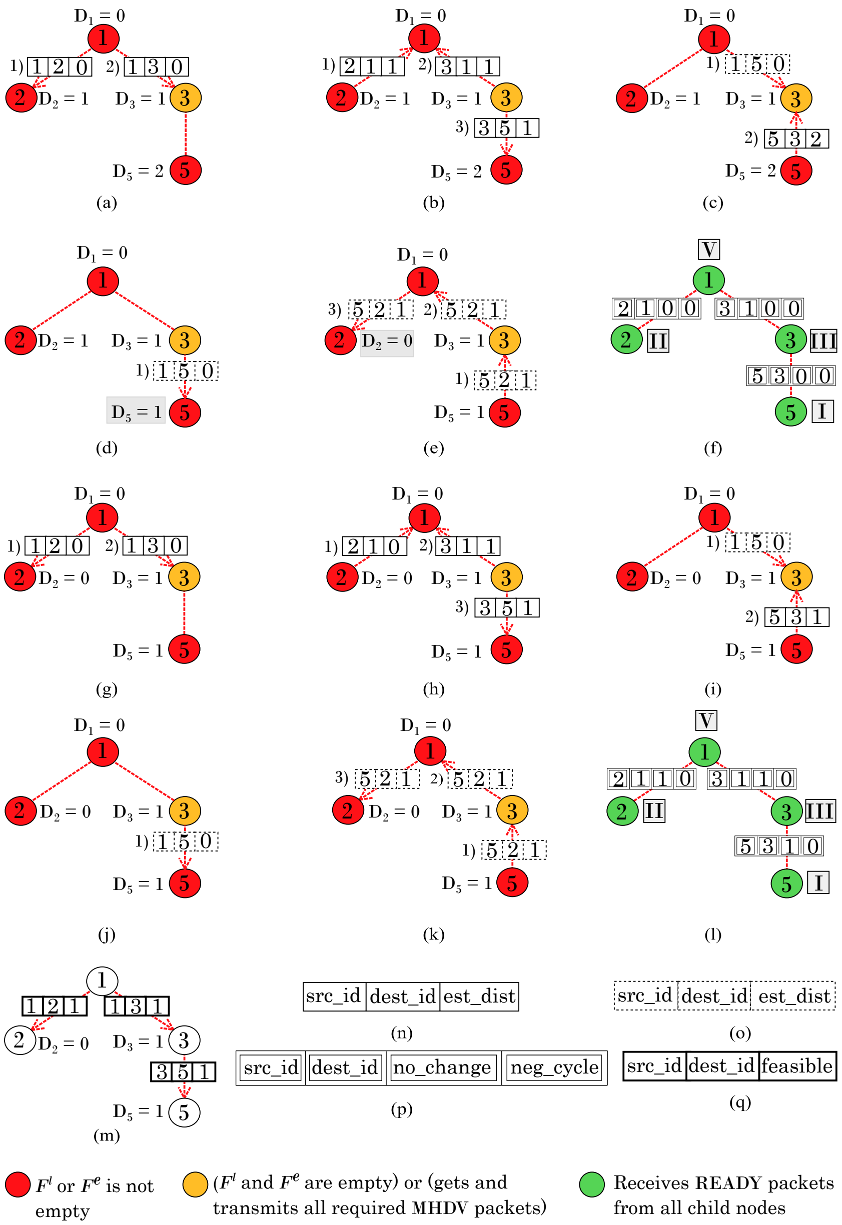


Figure 5.8: Distributed calculation of D_i for the example shown in Fig. 5.7, (a)…(f) the first iteration of packet transmissions, (g)…(i) the second iteration of packet transmissions, (m) transmission of STOP packet, (n) the SHDV packet format, (o) the MHDV packet format, (p) the READY packet format, (q) the STOP packet format.

the algorithm (Alg. 8) iterates again for the new given length of the schedule period.

The distributed calculation of D_i for the example shown in Fig. 5.7 is shown step by step in Fig. 5.8. Notice also that the leaf nodes, node 4 and 6, are not part of the calculation. Notice also that by considering Fig. 5.8m, then $Succ_1 = \{3\}$ and $Pred_1 = \{2\}$.

5.7 Calculation of the Active Portion Duration and the Number of Tx and Rx Slots:

The active portion of each cluster is in proportion to the amount of data to be exchanged between the cluster-head and its child nodes. Thus, the proper setting of the cluster active portion length requires the knowledge of the amount of data to be exchanged within the cluster. Such knowledge can be acquired by extending the FLOW-INFO packet, explained in Sec. 5.6.1, by two more fields:

1. *sample_size*: it is set to the $sampleSize_{f_q}$.
2. *sample_ack* it is set to the $sampleACK_{f_q}$.

The pseudo-code for the τ_i duration calculation of each C_i is shown in Alg. 10. The calculation is following the ZigBee standards [9]. The duration of τ, measured in number of time-slots, is related to the given number of CSMA-CA slots and the number of Rx, Tx slots to be allocated. Each Rx, Tx slot includes in addition to the effective data, the IFS, eventual acknowledgment and re-transmissions (Fig. 5.9). IFS separates consecutive frames. In the case of the acknowledged transmissions (i.e., $sampleACK = 1$) the sender waits for the corresponding acknowledgment frame, at most ($macAWD$) time units. If an acknowledgment frame is received within this time, the transmission is considered successful. Otherwise, the data transmission and waiting for the acknowledgment are repeated up to the maximum of $macMFR$ times. If an acknowledgment frame is not received after $macMFR$ re-transmissions, the transmission is considered as failed. The number of slots for Tx, Rx required for the whole data transmission (data frame, IFS, eventual acknowledgment and re-transmissions) is shown in Alg. 10 lines 6 to 11. The *frm_size* is the size of transmitted frame including the data payload as given by the *sampleSize* parameter, MAC and PHY headers; the *rate* is the data rate measured in kbps.

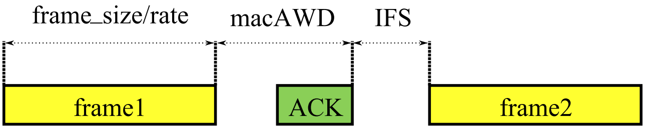


Figure 5.9: MAC frame.

Algorithm 10: The distributed calculation of the active portion and time-slots durations.

```
$\tau_i = \#Ts_{CSMA}$
$index = \#Ts_{CSMA}$
foreach $node\ j \in Child_i$ do
    foreach $FLOW\text{-}INFO\ pck_k\ received\ from\ node\ j$ do
        $\varphi_k = \left( \begin{array}{c} frm\_size_k/rate \quad + \\ macAWD \cdot pck_k.sample\_ack \end{array} \right) + IFS$
    $Tx_j = \sum_{pck_k} (macMFR \cdot pck_k.sample\_ack + 1) \cdot \varphi_k$
    $\#Tx_j = \left\lceil \frac{Tx_j}{t_s} \right\rceil$
    $I_{Tx_j} = index$   // index of first Tx slot of node $j$
    foreach $FLOW\text{-}INFO\ pck_l\ tranmitted\ to\ node\ j$ do
        $\varphi_l = \left( \begin{array}{c} frm\_size_l/rate \quad + \\ macAWD \cdot pck_l.sample\_ack \end{array} \right) + IFS$
    $Rx_j = \sum_{pck_l} (macMFR \cdot pck_l.sample\_ack + 1) \cdot \varphi_l$
    $\#Rx_j = \left\lceil \frac{Rx_j}{t_s} \right\rceil$
    $I_{Rx_j} = I_{Tx_j} + N_{Tx_j}$   // index of first Rx slot of node $j$
    $index = I_{Rx_j} + \#Rx_j$
    $\tau_i = \tau_i + (\#Tx_j + \#Rx_j)$
```

5.8 Cluster Offset Within the Schedule Period

At this stage, each C_i determines its offset, denoted by s_i, within the schedule period. The value of s_i indicates the first allocated time-slot to C_i. To determine the value of s_i, we propose two approaches. The first approach is utilized by DTDMA$^{\text{scd}}$ and considers the case in which , at most, one cluster can be active at any given time. The second approach is utilized by DTDMA$^{\text{mcd}}$, and supports the spatial reuse of the transmission medium in order to improve the bandwidth utilization particularly for the large-scale networks. However, the complexity of the problem with the consideration of the spatial reuse of the transmission medium while minimizing the total number of the allocated time-slots is an $\mathcal{NP}$-hard problem as proven in [22]. The proof is based on the reduction of the graph coloring problem, a well known $\mathcal{NP}$-hard problem, to the collision-free TDMA scheduling problem.

5.8.1 Single-Collision Domain Case Study

In such a case, one cluster, at most, is active at any given time. Hence, the active portion of the clusters fits into the schedule period provided that $\sum_{i \notin L} \tau_i \leq P$. To determine the value of s_i, the following constraints must hold:

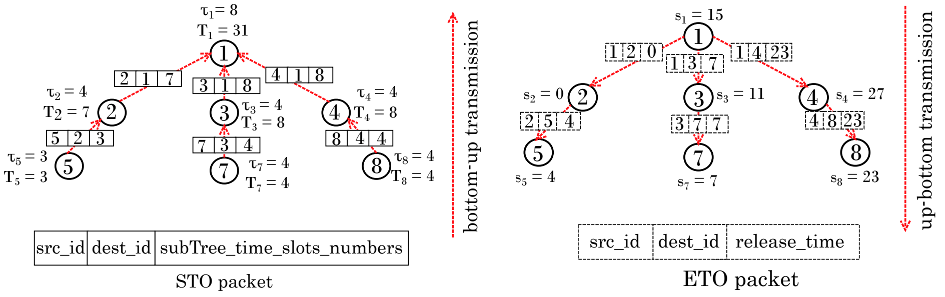
1. Each C_i is active for τ_i time-slots so that $s_i + \tau_i \leq P$.
2. The precedence decisions as given by the edges in POCA graph. Hence, for given $e(C_i, C_j) \in E(\text{POCA})$, $s_i + \tau_i \leq s_j$ must hold.
3. The collision avoidance constraint (i.e., allocating non overlapping time-slots for the clusters). Hence, for every C_i and C_j, one of the following constraints must hold: $s_i + \tau_i \leq s_j$ or $s_j + \tau_j \leq s_i$.

Since the POCA graph is a DAC, one topological ordering, at least, exists for the clusters as represented by the set of nodes $V(\text{POCA})$. Hence, by considering the POCA graph, the time-slots allocation problem in the case of a single-collision domain Cluster-Tree topology can be solved in polynomial time. Considering the POCA graph shown in Fig. 5.6b, the following order of the active portions of the clusters $(C_2, C_5, C_7, C_3, C_1, C_8, C_4)$ indicates that $s_2 = 0, s_5 = s_2 + \tau_2, \ldots, s_4 = s_8 + \tau_8$. In this section, we propose a distributed topological ordering algorithm of the clusters in the POCA graph.

The distributed topological ordering encompasses two phases where only the cluster-head nodes are involved. During the first phase, each node $i \notin L$ calculates the sum of τ_k of all nodes $k \in subT_i : k \notin L$. This sum is denoted by T_i. To accomplish the T_i calculation, each node $i \notin L$ receives Start Topological Ordering (STO) packet from each node $j \in Child_i : j \notin L$, calculates the value of T_i as shown in Eq. (5.7), and then sends STO$(i, parent_i, T_i)$ packet to the $parent_i$ node. This phase lasts until node 1 receives STO packets from its child nodes. The calculation of T_i for each C_i in addition to the transmission of the STO packets, while considering the POCA graph shown in Fig. 5.6b, are illustrated in Fig. 5.10a. The τ_i duration is also depicted next to each node while assuming $\#TS_{csma} = 2$ time-slots.

$$T_i = \sum_{j \in Child_i} T_j + \tau_i : i, j \notin L \tag{5.7}$$

During the second phase, each node $i \notin L$, computes the value of s_i using Eq. (5.8) and then transmits one packet, denoted by End Topological Ordering (ETO), to each node $j \in Child_i : j \notin L$. The ETO packet includes, in addition to the the *src_id* and *dest_id* fields, the *release_time* field that is set to the earliest time-slot that can be allocated for the active portion of $C_j : j \in Child_i$, denoted by r_j, so that $s_j \geq r_j$. The value of r_j is calculated



(a) The transmission of STO packets. (b) The transmission of ETO packets.

Figure 5.10: Distributed topological ordering during the calculation of the cluster offset stage of DTDMA$^{\text{scd}}$.

as shown in Eq. (5.9a), (5.9b) and (5.9c). Fig. 5.10b shows the value of s_i for each C_i in addition to transmissions of the ETO packets. The format of both STO and ETO packets is shown in Fig. 5.10c and d, respectively.

$$s_i = \sum_{j \in Pred_i} T_j + r_i : i, j \notin L \tag{5.8}$$

$$r_j = \begin{cases} 0 & \text{if } j = 1 & (5.9\text{a}) \\ r_i + \sum\limits_{\substack{k \in Pred_i \\ k < j}} T_k & \text{if } j \in Child_i \cap Pred_i & (5.9\text{b}) \\ s_i + \tau_i + \sum\limits_{\substack{k \in Succ_i \\ k < j}} T_k & \text{if } j \in Child_i \cap Succ_i & (5.9\text{c}) \end{cases}$$

Notice that the calculation of s_i based on the topological ordering of the POCA graph so that the active portions of the clusters do not overlap is an exact solution in the case of a single-collision domain Cluster-Tree topology. Moreover, such an approach is a heuristic approach when the active portions of the clusters might overlap (i.e., in the case of multiple-collision domains Cluster-Tree topology). However, such a heuristic is incapable of finding a solution when the length of the schedule period is so tight (e.g., when $\sum_{i \notin L} \tau_i > P$).

5.8.2 Multiple-Collision Domains Case Study

Considering the spatial reuse of the transmission medium is crucial for large scale WSNs. However, in such case, the collision-free time-slot allocation while minimizing the total number of allocated time-slots is an $\mathcal{NP}$-hard problem. Hence, at this stage, a heuristic approach is considered.

Algorithm 11: The centralized algorithm utilized over the cluster offset calculation.

```
1  foreach $C_i \in V(POCA)$ do
2      $r_i \leftarrow 0$
3  $Schedule \leftarrow \emptyset$
4  $\mathcal{C} \leftarrow getReleasedClusters(POCA)$
5  while $\mathcal{C} \neq \emptyset$ do
6      $C_x \leftarrow getClusterWithMaxID()$
7      $s_x \leftarrow$ call Alg. 12
8      $Schedule \leftarrow Schedule \cup (C_x, s_x, s_x + \tau_x)$
9      foreach $e(C_x, C_i) \in E(POCA)$ do
10         $r_i \leftarrow max\{r_i,\ s_x\ + \tau_x\}$
11     $remove\ C_x\ and\ E^+(C_x)\ from\ POCA$
12     $\mathcal{C} \leftarrow getReleasedClusters(POCA)$
13     return $Schedule$
```

The time-slot allocation problem at this stage imposes the first two constraints as shown in Sec: 5.8.1. While the third constraint must hold for the clusters that are in collision only (i.e., C_i and C_j might overlap if they are not in collision).

A similar problem was tackled in [53] where the authors considered a distributed aggregation scheduling algorithm (i.e., the child nodes are followed by their parent node in the schedule). Hence, a particular case of precedence relations are considered. Moreover, the authors assume an equal demand of time-slots for each node. Both previous assumptions lead to a more specific problem. However, our proposed approach can handle arbitrary precedence relations and arbitrary demands of the time-slots as given by the payload of the data flows. Before explaining the distributed algorithm utilized at this part, we propose a centralized counterpart algorithm, that can be distributed without imposing an extra calculation overhead when compared with the centralized algorithm proposed in [7].

The pseudo-code of the centralized algorithm is shown in Alg. 11. For each C_i, the release time r_i is set to 0. The value of r_i represents the earliest time-slots that might be allocated to C_i. Then the algorithm iterates until s_i is calculated by each C_i. At each iteration, the set of the released clusters is determined by the function *getReleasedClusters*(). The cluster is considered to be released when all of its predecessor clusters, as given by the POCA graph, are scheduled. Then, the function *getClusterWithMaxID*() returns the released cluster with the maximum *ID*, denoted by C_x in Alg. 11 at line 6. The Alg. 12 determines the value of s_x so that each C_x is allocated the earliest sufficient, consecutive and collision-free set of time-slots so that $s_x \geq r_x$. Once C_x is scheduled, the successor clusters to C_x, as given by the POCA graph, update their release time as shown at line 10. Moreover, C_x is removed from the POCA graph with all outgoing edges.

Algorithm 12: The centralized cluster offset adjustment algorithm.

```
$s_x \leftarrow r_x$
$Sch \leftarrow getScheduledCompetitors()$
$Sch \leftarrow sort(Sch)$  // sort scheduled competitors in ascending order based on the start time
foreach $C_j \in Sch$ do
    if $s_x + \tau_x \leq s_j$ then
        return $s_x$
    else
        $s_x \leftarrow max(s_x, s_j + \tau_j)$
return $s_x$
```

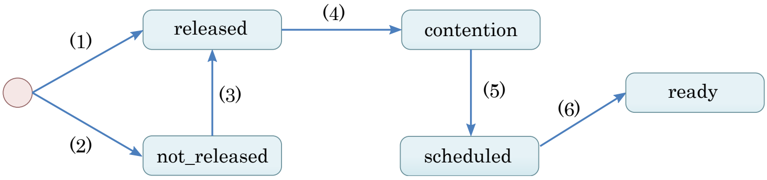


Figure 5.11: The cluster-head node finite state transition diagram during the calculation of the cluster offset stage of DTDMA^{mcd} algorithm. (1): $Pred = \emptyset$, (2): $Pred \neq \emptyset$, (3): $Pred = \emptyset$, (4): All PROBE-NOTIFY packets are sent, (5): all feedbacks are received and the current node has the maximum ID among the nodes in *Rel* set, (6): all READY packets have been received from all child nodes and the node has sent READY packet to its parent node.

To implement the counterpart distributed algorithm, we distinguish between five states of the cluster-head nodes as shown in Fig. 5.11. Recall that only the cluster-head nodes are involved at this stage. The state *released* indicates that all the predecessor clusters have been scheduled (i.e., all predecessor nodes as given by *Pred* set have determined their offset within the schedule period). Otherwise, the node is in *not-released* state. The node is in the *contention* state if it is released but not scheduled yet. When the node determines its offset within the schedule period, it gets into *scheduled* state. The scheduled node becomes READY when it receives READY packets from all its child nodes. The pseudo-code of the algorithm is shown in Alg. 13 so that the following parameters are maintained by each cluster-head node $i \notin L$:

1. M_i: this parameter is considered as an input and it includes the *ids* of the competitor clusters to C_i. Thus $M_i = \{j : C_j \in M(C_i)\}$.
2. st_i: the current state of node $i \notin L$. The node state is either *not-released, released, contention, scheduled or ready.*
3. $NRel_i$: the *ids* of the competitor clusters of C_i that are not released.

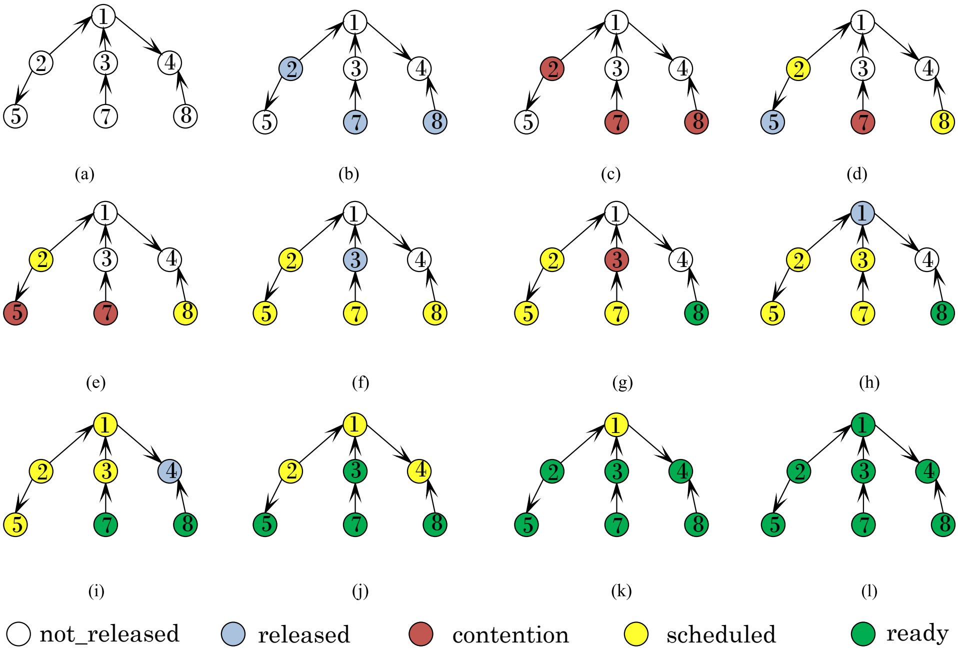

(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)

Figure 5.12: The changes on the cluster-head nodes states while applying the calculation of the cluster offset stage of DTDMA$^{\mathrm{mcd}}$ for the POCA graph shown in Fig. 5.6b

Thus, $NRel_i = \{j \in M_i \colon st_j = \textit{not-released}\}$.

4. Rel_i: the *ids* of the competitor clusters of C_i that are released but not scheduled yet. Hence $Rel_i = \{j \in M_i \colon st_j = \textit{released or } st_j = \textit{contention}\}$.
5. Sch_i: the *ids* of the competitor clusters of C_i that have been scheduled. Thus, $Sch_i = \{(s_j, s_j + \tau_j) \colon j \in M_i \textit{ and } st_j = \textit{scheduled}\}$.

The *setState*() function sets the node state to either *released* or *not-released* based on the *Pred* set. Once node $i \notin L$ becomes *released*, it sends PROBE-NOTIFY to its competitor nodes as given by M_i set and updates its state to *contention*. When an unscheduled node receives PROBE-NOTIFY, then as shown in Alg. 13 lines 8 to 13, the node firstly includes the prober into its *Rel* set and then it replies by sending RESPONSE to the sender. Through the RESPONSE packet, the responded node informs the prober node about its current state. As shown at lines 14 to 18, if the state of the responded node is either *released* or *contention*, the prober node, node i, includes the responded node into Rel_i set; otherwise it is included into $NRel_i$ set. The variable fb_i is used to store the number of the responses (i.e., feedbacks) the node i received so far. When the prober node receives all the feedback from its competitors (i.e., $fb_i = \mid M_i \mid$), then as shown in

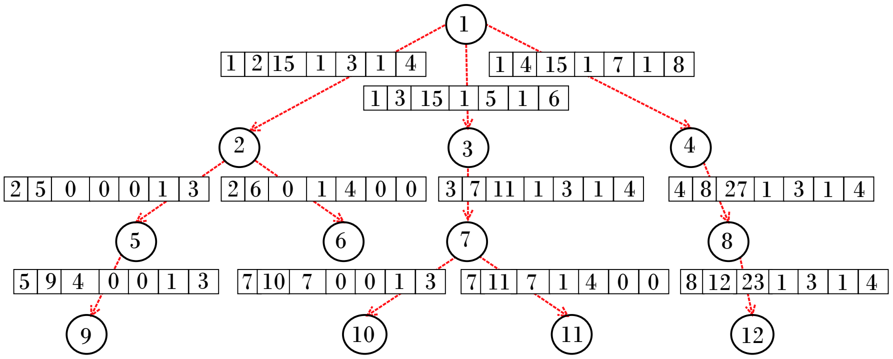
Algorithm 13: The distributed algorithm utilized over the cluster offset calculation.

```
1  $quit_i \leftarrow false$, $st_i \leftarrow setState()$, $r_i \leftarrow 0$, $fb_i \leftarrow 0$, $Rel_i \leftarrow \emptyset$, $NRel_i \leftarrow \emptyset$, $Sch_i \leftarrow \emptyset$,
     $ch_i \leftarrow 0$
2  while $\neg quit_i$ do
3      if $st_i \leftarrow released$ then
4          send PROBE-NOTIFY packet to each node $v \in M_i$
5          $st_i \leftarrow contention$
6      $pck \leftarrow$ receive packet from node $j \in M_i \cup Child_i$
7      switch $pck.type$ do
8          case PROBE-NOTIFY do
9              if $st_i \neq scheduled$ then
10                 $Rel_i \leftarrow Rel_i \cup \{j\}$
11                 $NRel_i \leftarrow NRel_i \setminus \{j\}$
12                 send PROBE-RESPONSE packet to the sender
13             break
14         case PROBE-RESPONSE do
15             $fb_i \leftarrow fb_i + 1$
16             if $st_j = contention$ then
17                 $Rel_i \leftarrow Rel_i \cup \{j\}$
18             break
19         case SCHED-DETER do
20             $Sch_i \leftarrow Sch_i \cup (s_j, s_j + \tau_j)$
21             if $st_i = contention$ then
22                 if $j \notin Rel_i \cup NRel_i$ then
23                     $fb_i \leftarrow fb_i + 1$
24                 $Rel_i \leftarrow Rel_i \setminus \{j\}$
25             if $st_i = not\text{-}released$ then
26                 $M_i \leftarrow M_i \setminus \{j\}$
27                 if $j \in Pred_i$ then
28                     $Pred_i \leftarrow Pred_i \setminus \{j\}$
29                     $r_i \leftarrow max\{r_i, s_j + \tau_j\}$
30                     $st_i \leftarrow setState()$
31             break
32         case READY do
33             $ch_i \leftarrow ch_i + 1$
34             break
35     if $st_i = contention$ and $fb_i = \mid M_i \mid$ then
36         if $i > max\{j : j \in Rel_i\}$ then
37             $s_i \leftarrow$ call Alg. 14
38             $st_i \leftarrow scheduled$
39             send SCHED-DETER packet to each node $v \in M_i$
40     if $st_i = scheduled$ and $ch_i = \mid Child_i \mid$ then
41         $st_i \leftarrow$ READY
42         send READY packet to $parent_i$
43         $quit_i \leftarrow true$
```

Alg. 13 at lines 35 to 39, it checks whether it has the greatest *ID* among the nodes included into Rel_i set. If so, it calls Alg. 14 to allocate the earliest sufficient, consecutive and collision-free set of time-slots while considering all the scheduled competitors that are included into the *Sch* set. Then the scheduled node sets its state to *scheduled* and sends SCHED-DETER to all nodes included into M_i set. When a node receives SCHED-DETER

Algorithm 14: The distributed cluster offset adjustment algorithm.

```
$s_i \leftarrow r_i$
$Sch_i \leftarrow sort(Sch_i)$   // sort scheduled competitors in ascending order based
    on the start time
foreach $j \in Sch_i$ do
    if $s_i + \tau_i \leq s_j$ then
        return $s_i$
    else
        $s_i \leftarrow max\{s_i, s_j + \tau_j\}$
return $s_i$
```



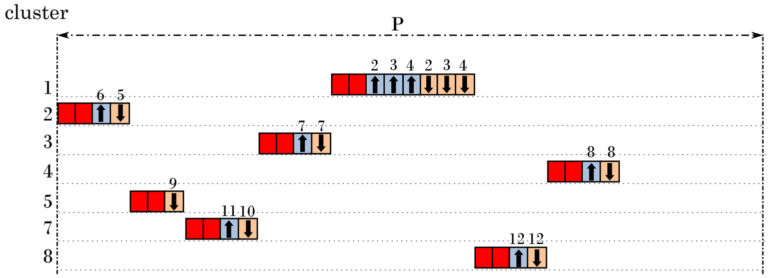
src_id	dest_id	start_time	num_Tx_slots	first_Tx_slot_index	num_Rx_slots	first_Rx_slot_index

TSA packet

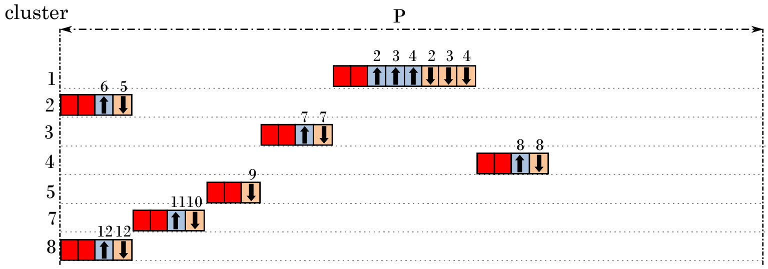
Figure 5.13: The *parent-child* time-slots allocation stage for the example shown in Fig. 5.10

packet, as shown at lines 19 to 31, two cases might be distinguished. The first case arises when the receiver is in *contention* state, then it removes the scheduled node from its *Rel* set and includes it in the *Sch* set. The second case arises when the receiver is in *not-released* state. In such a case, the receiver removes the scheduled node from its competitor set and checks whether all its predecessors nodes have been scheduled. If so, it sets its state to *released* and sends PROBE-NOTIFY packets to its competitors. The READY packets, as shown in Alg. 13 at lines 40 to 43, are utilized at this stage to indicate that the node has completed its duties at this stage. In other words, the scheduled node sends READY packets to its parent node when all the nodes that belong to the subtree rooted by that node, as given in Fig. 5.2, are scheduled.

Fig. 5.12 illustrates step by step the proposed distributed algorithm shown in Alg. 13 when the POCA graph illustrated in Fig. 5.6b is considered as input to the algorithm.



(a) cluster schedule as returned by DTDMA$^{\text{scd}}$.



(b) cluster schedule as returned by DTDMA$^{\text{mcd}}$.

Figure 5.14: A cluster schedule of the example shown in Fig. 5.1.

5.9 Parent-child time-slot Allocation

As explained in Sec. 5.3.3, each cluster-head allocates *Tx* and *Rx* time-slots to its child nodes in order to exchange the data within the cluster. This stage of the algorithm, at each node $i \notin L$, commences when node i calculates the value of s_i. Thus, each node i sends one packet, denoted by Time-Slot Allocation (TSA), to each node $j \in Child_i$. The packet includes the following fields in addition to *src_id* and *dest_id*:

1. *start_time*: this field is set to the value of s_i.
2. *num_Tx_slots*: it is set to the number of the *Tx* time-slots to be allocated by node i to node j as denoted by $\#Tx_j$ in Sec. 5.7.
3. *first_Tx_slot_index*: it is set to the index of the first *Tx* time-slot to be allocated by node i to node j as denoted by I_{Tx_j}.
4. *num_Rx_slots*: it is set to the number of the *Rx* time-slots to be allocated by node i to node j as denoted by $\#Tx_j$.
5. *first_Rx_slot_index*: it is set to the index of the first *Rx* time-slot allocated by node i to node j as denoted by I_{Rx_j}.

Each child node j utilizes the received *TSA* packet to synchronize its assigned *Tx* and *Rx* time-slots with its parent (i.e., the child node j sends the data to its parent node i during the time-slots interval given by $[s_i + I_{Tx_j}, s_i + I_{Tx_j} + \#Tx_j]$ and receives data from its parent node during the time-slots interval given by $[s_i + I_{Rx_j}, s_i + I_{Rx_j} + \#Rx_j]$. The transmissions of the *TSA* packet while considering the example shown in Fig. 5.10 is shown in Fig. 5.13.

In Fig. 5.14, we show a cluster schedule solution for the example shown in Fig. 5.1 in both cases of single and multiple-collision domains.

5.10 Experimental Results

To carry out the experimental results, $\mathrm{DTDMA}^{\mathrm{scd}}$, $\mathrm{DTDMA}^{\mathrm{mcd}}$, and the distributed scheduling algorithm proposed in [3] for a single-collision domain, denoted by $\mathrm{DTDMA}^{\mathrm{s}}$, are implemented in Java.

$\mathrm{DTDMA}^{\mathrm{s}}$, $\mathrm{DTDMA}^{\mathrm{scd}}$ and $\mathrm{DTDMA}^{\mathrm{mcd}}$ are evaluated with respect to the average number of returned feasible schedules (i.e., the success rate), the overhead in terms of the number of transmitted packets and energy consumption, and the elapsed time up to the time instant at which the schedule is constructed (i.e., the computation time). Moreover, we compared our proposed algorithms with TDCS approach, that is based on ILP [25], in terms of the success rate and the computation time. The proposed ILP model can be utilized to construct a compact schedule (i.e., a model with an objective function that minimizes the makespan of the schedule) or a feasible schedule (i.e., a model with no objective function). To solve the model, we use Gurobi solver with a time limit set to 10 min.

Also, we compare the energy efficiency and the network reliability as based on the schedule obtained by $\mathrm{DTDMA}^{\mathrm{mcd}}$ with the schedule obtained by the TWBS approach that is proposed in [51].

5.10.1 Benchmarks Settings

Each benchmark, one row in Tab. 5.4, consists of 30 instances where each problem instance is composed of a Cluster-Tree topology and a set of user defined data flows.

The number of nodes and clusters in each Cluster-Tree topology are illustrated in the first column in Tab. 5.4. For each Cluster-Tree topology, the root node is placed in the center of the square of size (2000×2000) m^2 while the other nodes are distributed so that each cluster-head may have 6 child nodes (3 child cluster-head nodes and 3 leaf nodes). The transmission and the carrier sensing area for each node are given by 25 m and 40 m respectively. The competitor set of each cluster as denoted by $M(C_u)$, is determined as given by Eq. 5.2 based on the coordinates of the nodes (i.e., physical deployment) together with the carrier sensing area parameter set to 40 m.

The user defined data flows are described by the number of data flows

#flows, the number of sources #src, the *e2eDeadline* and the *reqPeriod* parameters (see columns $2 - 4$). The *reqPeriod* is set so that the minimum length of the schedule period is long enough to fit the active portion of the clusters when the *e2eDeadline* parameter is assumed to be infinity. In other word, considering the active portion duration of the clusters and the competitor set of each cluster, the compact cluster schedule, that minimizes the completion time of the schedule (i.e., the schedule makespan) as denoted by C_{max}, is found. Then, PO_{min} is set to the smallest value so that the resulting $P_{min} \geq C_{max}$. Thus, the *reqPeriod* is set to P_{min}. Then, the *e2eDeadline* parameter is set so that *e2eDeadline* $>$ *reqPeriod*. Moreover, for each benchmark, the corresponding PO_{max}, as given by the *reqPeriod* parameters (see Eq. (5.1)), is illustrated in column 5.

To study the impact of the *e2eDeadline* and the *reqPeriod* parameters, (more precisely, the maximum number of crossed periods of each data flow), on the performance of the considered algorithms, then for every two consecutive benchmarks, the value of the *e2eDeadline* is increased so that the maximum number of crossed periods for each data flow is increased. Thus, for the successive benchmarks with an equal value of *reqPeriod* parameter, the *e2eDeadline* is increased by an integer number of the *reqPeriod*. When all algorithms solves all instances in a given benchmark, the *reqPeriod* is increased so that PO_{max} is increased by one and so on.

We assume, without loss of generality, that the *sampleSize* $= 64$ bits; and *sampleACK* $= 0$. The transmission rate is assumed to be 250 kbps while considering a frequency band of 2.4 GHz. Due to different kinds of disturbances, the transmission over the channel might be lost in real case WSNs. Thus, and for the sake of simplicity, we assume that the channel error rate, denoted by *loss_rate*, is fixed so that *loss_rate* $\in \{30\%, 40\%, 50\%\}$. Hence, the node drops the received packet with the probability of 0.3, 0.4 or 0.5.

In Tab. 5.4 and Fig. 5.15, we assumed that the *loss_rate* $= 30\%$ and the clusters are distributed on the field such that the resulting average number of competitor clusters in each benchmark, denoted by $avg(|M(C_u)|)$, is given by $avg(|M(C_u)|) \leq 25$.

5.10.2 Success Rate of the Algorithms

The success rate of the evaluated algorithms is computed as the percentage of the average number of the returned feasible schedules per each set of instances in each benchmark. As shown in the columns *succ_rate* in Tab. 5.4 and in Fig. 5.15a, the *succ_rate* of DTDMA$^{\mathrm{mcd}}$ is higher than the *succ_rate* of DTDMA$^{\mathrm{scd}}$. In principle, DTDMA$^{\mathrm{scd}}$ fails to find a schedule when the length of the schedule period is less than the sum of the active portion of the clusters. On the other hand, since DTDMA$^{\mathrm{mcd}}$ utilizes the spatial reuse of the transmission medium, in comparison with DTDMA$^{\mathrm{scd}}$, then even when the schedule period is so tight, DTDMA$^{\mathrm{mcd}}$ might succeed in fitting the active portions of the clusters within the schedule period.

To realize the impact of the *reqPeriod* and *e2eDeadline* on the success rate of the algorithms, consider the case of 400 clusters and $\#flows = 40$. When the $reqPeriod = 10$ s (i.e., $P = P_{max} = 15.7$ s) and $e2eDeadline = 80$ s, then $succ\_rate = 33.3\%$ for DTDMA$^{\mathrm{scd}}$ and $succ\_rate = 100\%$ for DTDMA$^{\mathrm{mcd}}$. Such behavior can be explained as follows: when $P = P_{max} = 15.7$ s and for some instances, both DTDMA$^{\mathrm{mcd}}$ and DTDMA$^{\mathrm{scd}}$ fails during the construction of the POCA graph due to the existence of a negative cycle in the inequality graph. To eliminate the negative cycle, both DTDMA$^{\mathrm{mcd}}$ DTDMA$^{\mathrm{scd}}$ decreases the length of the schedule period in order to increase the maximum number of crossed periods for each data flow. Thus, by setting $P = 7.86$ s, the negative cycle problem is resolved, however, DTDMA$^{\mathrm{scd}}$, in contrast to DTDMA$^{\mathrm{mcd}}$, fails to fit the active portions of the clusters into the given $P = 7.86$ s. Recall that, DTDMA$^{\mathrm{scd}}$ requires that the length of the schedule period is not shorter than the sum of the active portion of the clusters. Notice also, by setting $P = P_{max} = 15.7$ s and $e2eDeadline = 96$ s, the $succ\_rate = 100\%$ for both DTDMA$^{\mathrm{scd}}$ and DTDMA$^{\mathrm{mcd}}$.

Considering the comparison with DTDMA$^{\mathrm{s}}$, and since both DTDMA$^{\mathrm{scd}}$ and DTDMA$^{\mathrm{s}}$ are exact algorithms for the case of a single-collision domain, then both algorithms return schedules with equal makespan that is equal to the sum of the active portions for all clusters. Consequently both algorithms have an equal *succ_rate*. However, DTDMA$^{\mathrm{scd}}$ has a better performance than DTDMA$^{\mathrm{s}}$ in terms of computation time, the number of transmitted packets, and the energy consumption as will be explained in Sec. 5.10.3, Sec. 5.10.4, and Sec. 5.10.5, respectively.

The success rate of the TDCS is also illustrated in Fig. 5.15a. Notice that, the success rate of the TDCS decreases as the number of clusters increases due to the complexity of solving the ILP model (i.e., the success rate equals 0% in the case of a compact schedule when the number of clusters reaches 200 and the success rate is less than 40% in the case of a feasible schedule when the number of clusters reaches 600). Notice also that for instances up to 200 clusters, the TDCS outperforms DTDMA$^{\mathrm{mcd}}$ in returning a feasible schedule. This behavior is due to the fact that expressing the *e2eDeadline*, as the maximum number of the crossed periods, rather than in seconds as in TDCS, leads to an elegant approach with shorter computation time for DTDMA$^{\mathrm{mcd}}$ as will be discussed in Sec. 5.10.3, on the cost of the inability to find a feasible schedule for some instances. However, the aforementioned cost is eliminated when the *e2eDeadline* is increased.

Table 5.4: The evaluation of $\text{DTDMA}^{\text{scd}}$ and $\text{DTDMA}^{\text{mcd}}$ algorithms when $loss\_rate = 30\%$ and $avg(|C_m|) \leq 25$.

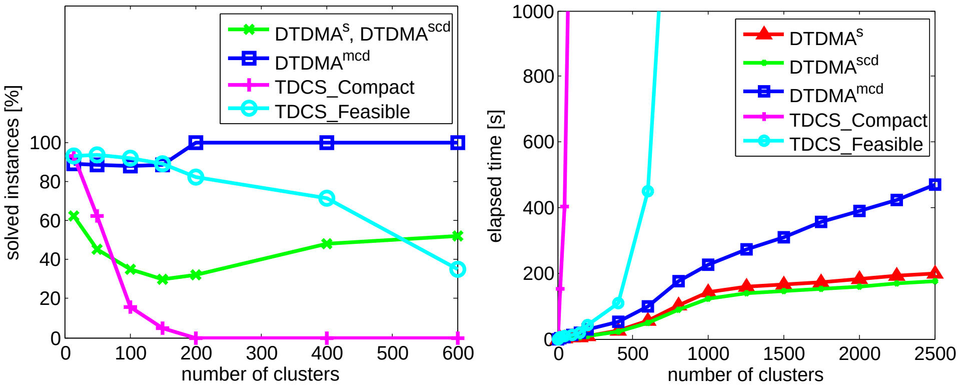
Problem Instance					$\text{DTDMA}^{\text{scd}}$				$\text{DTDMA}^{\text{mcd}}$			
#nodes (#clusters)	#flows (#src)	e2eDeadline [s]	reqPeriod [s]	PO_{max}	$succ\_rate$ [%]	t_{feas} [s]	$avg(pck)$	$avg(E)$ [mJ]	$succ\_rate$ [%]	t_{feas} [s]	$avg(pck)$	$avg(E)$ [mJ]
60 (15)	3 (4)	∞	0.25	4	0	–	–	–	100	0.38	45.8	8
		1	0.5	5	70	0.367	46.1	8	100	0.494	61.0	11
		1.5	0.5	5	100	0.273	33.6	6	100	0.407	49.4	9
		2	0.5	5	100	0.251	30.43	6	100	0.389	46.4	8
	5 (4)	∞	0.25	4	0	–	–	–	46.7	0.482	56.27	7
		1	0.5	5	36.7	0.520	61.7	11	66.7	0.665	80.03	14
		1.5	0.5	5	93.3	0.413	47.73	9	96.7	0.568	67.0	12
		2	0.5	5	100	0.330	37.7	7	100	0.476	56.06	10
600 (150)	20 (4)	∞	2	7	0	–	–	–	100	18.3	246.9	41
		8	4	8	0	–	–	–	26.7	22.4	329.56	55
		12	4	8	0	–	–	–	100	20	265.56	44
		16	4	8	26.7	7.6	107.5	18	100	19.9	309.9	52
		20	4	8	100	4.69	46.7	8	100	17.4	248.96	42
		40	8	9	100	4.96	46.70	8	100	17.65	248.96	42
	30 (4)	∞	2	7	0	–	–	–	76.7	18.39	257.4	43
		12	4	8	0	–	–	–	83.3	19.79	285.93	48
		16	4	8	0	–	–	–	86.7	22.6	334.46	56
		20	4	8	0	–	–	–	100	19.94	274.5	46
		40	8	9	100	6.04	63.36	11	100	19.6	275.7	46
1600 (400)	40 (4)	∞	8	9	0	–	–	–	100	49.5	388.96	65
		80	16	10	33.3	24.10	137.76	23	100	58.55	456.13	76
		96	16	10	100	15.196	46.6	8	100	47.95	367.3	62
	60 (4)	∞	8	9	0	–	–	–	100	52.631	399.53	67
		80	16	10	3.3	29.681	180.833	30	100	62.35	500.6	83
		96	16	10	100	17.97	56.86	10	100	50.943	375.36	63
		112	16	10	100	15.39	42.16	8	100	47.77	360.067	61
4000 (1000)	100 (4)	160	32	11	0	–	–	–	100	141.8	471.6	79
		320	64	12	100	78.4	85.6	15	100	140.2	469.05	79
		384	64	12	100	88.2	142.75	24	100	156.97	509.2	85
	150 (4)	320	64	12	95	147.78	99.35	18	100	303.17	808.3	136
		384	64	12	95	172.30	334.45	56	100	331.23	1044	174
		448	64	12	100	124.56	46.3	9	100	278.58	755.85	127

5.10.3 Computation Time of the Algorithms

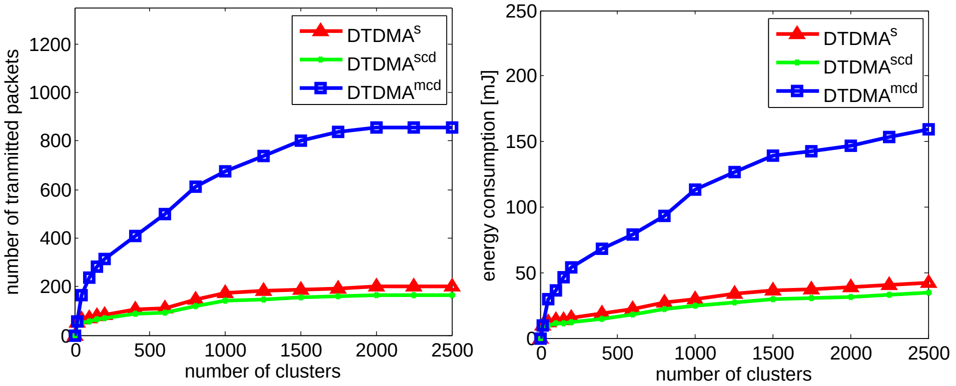
In this section, we demonstrate the overhead of the evaluated algorithms in terms of the required time to construct the schedule. In other words, we focus on the time elapsed between the time instant at which nodes start running the scheduling algorithm and the time instant at which the network becomes functional with respect to the constructed schedule.

The computation time (i.e., the elapsed time) for the feasible instances for each benchmark, is shown in column t_{feas} in Tab. 5.4 for both $\text{DTDMA}^{\text{scd}}$ and $\text{DTDMA}^{\text{mcd}}$ algorithms. Moreover, the average computation time for DTDMA^{s}, $\text{DTDMA}^{\text{scd}}$, and $\text{DTDMA}^{\text{mcd}}$ algorithms per each set of clusters is shown in Fig. 5.15b.

The results show that even for instances with 2500 clusters, the average computation time in the case of $\text{DTDMA}^{\text{scd}}$ is less than 3 min while it is less



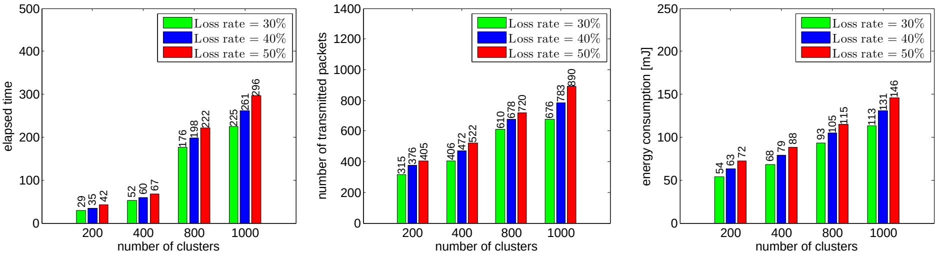
(a) Solved instances (b) Elapsed time



(c) Transmitted packets. (d) Energy consumption.

Figure 5.15: The evaluation of DTDMA$^{\mathrm{s}}$, DTDMA$^{\mathrm{scd}}$, DTDMA$^{\mathrm{mcd}}$ and TDCS algorithms when $loss\_rate = 30\%$ and $avg(|M(C_u)|) \leq 25$.

than 10 min in the case of DTDMA$^{\mathrm{mcd}}$. Hence, the average computation time for both algorithms is low. However, the average computation time of DTDMA$^{\mathrm{scd}}$ is smaller than the average computation time of DTDMA$^{\mathrm{mcd}}$ as shown in Fig. 5.15b. The reason lies behind the overhead of extra packets transmitted by DTDMA$^{\mathrm{mcd}}$, in comparison with DTDMA$^{\mathrm{scd}}$, during the calculation of the cluster offset stage as explained in Sec. 5.8. However, as explained in Sec. 5.10.2, the smaller computation time by DTDMA$^{\mathrm{scd}}$ is at the cost of the inability to find a feasible schedule when the length of the schedule period is shorter than the sum of the active portions of the clusters. Moreover, the results show, that the elapsed time keeps increasing for both algorithms as the number of clusters increases. Such increase is caused by the tree structure of underling Cluster-Tree topology together with the fact that both DTDMA$^{\mathrm{scd}}$ and DTDMA$^{\mathrm{mcd}}$ are multi-stages algorithms such that at each stage, READY packets are transmitted in bottom-up pattern towards the root of the network.



(a) Elapsed time. (b) Transmitted packets. (c) Energy consumption

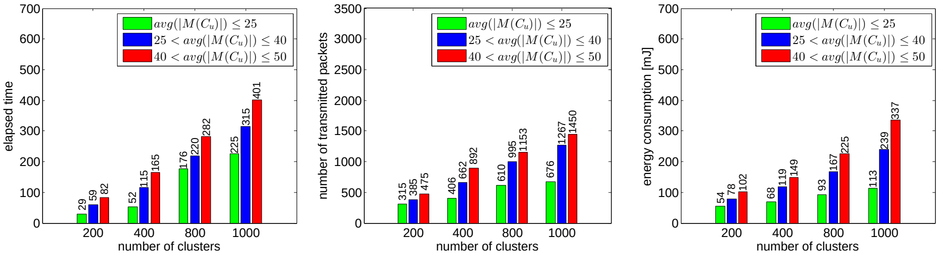
Figure 5.16: The performance evaluation of DTDMA$^{\text{mcd}}$ as a function to the *loss_rate* parameter.

The computation time for DTDMA$^{\text{s}}$ is also demonstrated in Fig. 5.15b. Notice that, the computation time of DTDMA$^{\text{scd}}$ is less than the computation time of DTDMA$^{\text{s}}$ which indicates the performance improvement achieved by DTDMA$^{\text{scd}}$. Such an improvement is due to the reduction in the length of the FLOW-INFO and FLOW-ACK packets and the reduction in the number of transmitted packets during the D_i calculations as explained in Sec. 5.6.1 and Sec. 5.6.2, respectively.

The computation time of the TDCS approach, that is based on ILP [25], is also depicted in Fig. 5.15b. The time limit for Gurobi solver is set to 10 min for both compact and feasible schedules. The results show that within the specified time limit of 10 min, the TDCS algorithm is able to construct a compact schedule for instances up to 30 clusters and a feasible schedule for instances up to 600 clusters. Hence, our proposed algorithms provide a solution for larger instances within a shorter time which proves the time efficiency of the algorithms proposed in this chapter.

Furthermore, in Fig. 5.16a, we show the elapsed time of DTDMA$^{\text{mcd}}$ as a function to the *loss_rate*. The higher the loss rate, the higher the probability that the transmission fails and, consequently, the sender re-transmits the packet. Hence, the elapsed time is increased.

Fig. 5.17a demonstrates the elapsed time of DTDMA$^{\text{mcd}}$ as a function to the number of competitor clusters as denoted by $|M(C_u)|$. Recall that, the number of competitor clusters, for a given cluster, depends on the physical deployment of the network and on the carrier sensing area of the clusters. Hence, when the clusters are distributed more closely to each other, the number of competitor clusters for each cluster is increased. Such an increase leads to more packets being transmitted by each cluster-head node during the calculation of the cluster offset stage of DTDMA$^{\text{mcd}}$ algorithm (i.e., the PROBE-NOTIFY, RESPONSE, and SCHED-DETER packets). Hence, the elapsed time is increased.



(a) Elapsed time. (b) Transmitted packets. (c) Energy consumption.

Figure 5.17: The performance evaluation of DTDMA$^{\mathrm{mcd}}$ as a function to the competitor set.

5.10.4 Number of Transmitted Packets

The average number of transmitted packets considering the feasible solutions returned by DTDMA$^{\mathrm{scd}}$ and DTDMA$^{\mathrm{mcd}}$, is shown in the avg(#pck) columns. Moreover, the average number of transmitted packets in the case of DTDMA$^{\mathrm{s}}$, DTDMA$^{\mathrm{scd}}$, and DTDMA$^{\mathrm{mcd}}$ per each set of clusters is shown in Fig. 5.15c.

The results show that the average number of packets transmitted by DTDMA$^{\mathrm{scd}}$ converges to 155 when the number of clusters reaches 1500 while the number of transmitted packets by DTDMA$^{\mathrm{mcd}}$ converges to 856 packets when the number of clusters equals 2000. However, as mentioned in Sec. 5.10.3, despite the convergence of the number of transmitted packets, the elapsed time keeps increasing when the number of clusters increases as shown in Fig. 5.15b. The results also show that DTDMA$^{\mathrm{mcd}}$ sends more packets in comparison with DTDMA$^{\mathrm{scd}}$. Moreover, by observing the difference in the average number of the transmitted packets between DTDMA$^{\mathrm{scd}}$ and DTDMA$^{\mathrm{mcd}}$, we can conclude that stage 4, the cluster offset calculation, is the stage at which most of the packets, roughly 65%, are transmitted when the nodes run DTDMA$^{\mathrm{mcd}}$ algorithm. Recall that stages $1, 2, 3, 5$ are identical in both algorithms. However, such an increase in the number of packets is inevitable when the usage of the spatial reuse of the transmission medium is crucial for better bandwidth utilization, on one hand, and for the ability to find a feasible schedule when the schedule period is so tight, on the other.

Furthermore, for benchmarks with an equal number of clusters, the results show that the number of transmitted packets increases when the number of data flows increases. Such an increase is due to the fact that by increasing the number of data flows, the number of packets transmitted during the determination of the precedence decisions stage is also increased. Essentially, there is an increase in the number of FLOW-INFO and FLOW-ACK during the creation of the inequality graph phase and consequently, an increase in the number of MHDV through the D_i calculation phase as explained in Sec. 5.6.1 and Sec. 5.6.2, respectively. However, since in the case of feasible shortest path problem (i.e., weighted graph without negative cycles), the actual sufficient number of iterations required to solve the shortest path tree problem might be decreased (i.e., either the same or less) when the weight of the edges is increased, then increasing the value of the *e2eDeadline* of each data flow so that the maximum number of crossed periods of each data flow is also increased, increases the weights on the dashed edges of the inequality graph and, consequently, the number of iterations required by the D_i calculation stage might be decreased (i.e., either the very same value or less). Hence, the number of transmitted packets within the D_i calculation phase might be decreased and consequently, the average number of transmitted packets by both DTDMA$^{\mathrm{scd}}$ and DTDMA$^{\mathrm{mcd}}$ is decreased. See, for example, in Tab. 5.4, the instances with 150 clusters and the case when the *e2eDeadline* is increased from 16 s to 40 s. On the other hand, in the case of the unfeasible shortest path tree problem, then increasing the weight of the edges while the problem is still unfeasible might lead to the case at which more iterations are required to detect the negative cycle in the graph during the D_i calculation phase which leads to more packets being transmitted during the D_i calculation phase. Consequently, the average number of transmitted packets for both DTDMA$^{\mathrm{scd}}$ and DTDMA$^{\mathrm{mcd}}$ is increased. See, in Tab. 5.4, the instances with 1000 clusters and the case when the *e2eDeadline* is increased from 320 s to 384 s.

Fig. 5.15c shows that DTDMA$^{\mathrm{scd}}$ transmits a smaller number of packets in comparison with DTDMA$^{\mathrm{s}}$. This improvement is achieved by excluding the leaf nodes during the D_i calculation stage and the cluster offset calculation stage. Thus, during these two stages, the cluster-head nodes are not sending any packet to the leaf nodes.

Fig. 5.16b shows the number of packets being transmitted by DTDMA$^{\mathrm{mcd}}$ as a function to the loss rate. When the packet is lost, the sender re-transmits that packet again until the receiver successfully receives the packet. The results show, as expected, that the higher the loss rate, the higher the number of transmitted packets.

Fig. 5.17b demonstrates the number of transmitted packets as a function to the number of competitor clusters as denoted by $|M(C_u)|$. When the number of competitor clusters for each cluster is increased, there is an increase in the number of packets being transmitted by each cluster-head node during the calculation of the cluster offset stage of DTDMA$^{\mathrm{mcd}}$ algorithm (i.e., the PROBE-NOTIFY, RESPONSE, and SCHED-DETER packets).

5.10.5 Energy Consumption

In this section, we demonstrate the energy consumption of the nodes during the calculation of the schedule by both DTDMA$^{\text{scd}}$ and DTDMA$^{\text{mcd}}$. Furthermore, we calculate the total energy consumption of the network when the clusters are active as determined by the acquired schedule for a duration of 60 min.

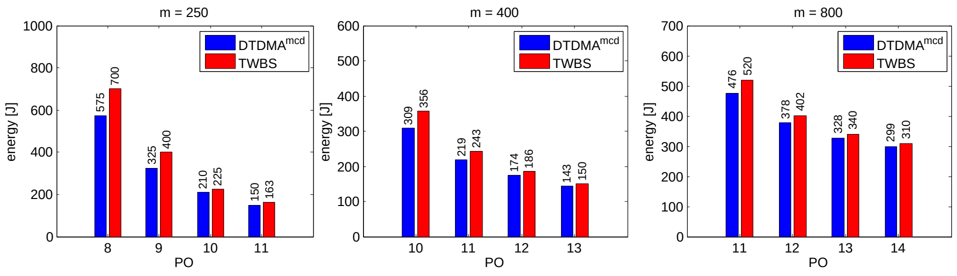
The energy consumption E is measured in $[J]$ and calculated by the formula: $E = U \cdot I \cdot t$ where U is the voltage, I is the current drawn and t is the execution time (i.e., transmitting or receiving times). The particular voltage is 3 V and the current drawns are given as follows: the current drawn in *receive mode* = 18.2 mA, *transmit mode* = 19.2 mA at 0 dBm, *idle mode* = 54.5 μA and *sleep mode* = 15 μA following ([7]). We assume a 2.4 GHz frequency band and 250 kbps of bit rate.

In this section, we demonstrate both the energy consumption of the scheduling algorithm and the energy consumption based on the resulting schedule.

The energy consumption of the cluster-heads are shown in columns avg(E) in Tab. 5.4 and in Fig. 5.15d. The results show that the energy consumption increases when the number of clusters increases within the network despite of the convergence of the number of packets as shown in Fig.5.15c due to the increase of the elapsed time as shown in Fig. 5.15b.

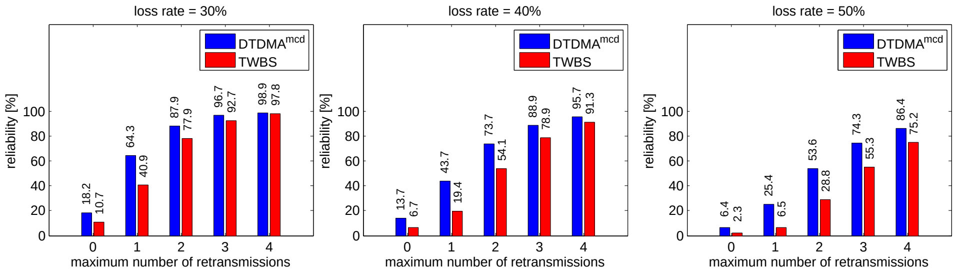
The higher overhead of DTDMA$^{\text{mcd}}$ in terms of the number of packets transmitted by the cluster-head nodes to construct the schedule in comparison with DTDMA$^{\text{scd}}$, leads to the case at which the cluster-head nodes consume more energy up to the time at which the schedule is constructed. However, even for instances with 2500 clusters, the average energy consumption per each cluster in the case of DTDMA$^{\text{scd}}$ is less than 50 mJ while it is less than 160 mJ in the case of DTDMA$^{\text{mcd}}$. Moreover, Fig. 5.15d also shows that DTDMA$^{\text{scd}}$ consumes less energy in comparison with DTDMA$^{\text{s}}$. Such a reduction in the energy consumption is due to two factors: The first one is the reduction in the number of transmitted packets as shown in Sec. 5.10.4 while the second one is the reduction in the length of the FLOW-INFO and FLOW-ACK packets.

Moreover, in Fig. 5.16c, we demonstrate the energy consumption as a function to the loss rate. The higher the loss rate, the higher the portability of the transmission failure occurrence. Hence, the sender re-transmits the packet and, thus, the energy consumption is increased. Furthermore, in Fig. 5.17c, we demonstrate the energy consumption as a function to the average number of competitors. The higher the number of the competitor clusters for each cluster, the higher the number of packets being transmitted over the cluster offset calculation stage of DTDMA$^{\text{mcd}}$ and, consequently,



(a) (b) (c)

Figure 5.18: The energy consumption of the network within 60 min based on the cluster-schedule obtained by both DTDMA^{mcd} and TWBS as a function to the value of PO and the number of clusters.



(a) (b) (c)

Figure 5.19: The impact of the loss rate on the network reliability

the higher the energy consumption.

The results prove the energy efficiency of our proposed distributed algorithms. Even though the number of transmitted packets might be high, nevertheless, the packets are of a small size and, consequently, the energy consumption is low.

we demonstrate the energy consumption of all clusters, as a function to the value of PO, when the cluster schedule is running for a duration of 60 min. Moreover, we compare our approach with the TWBS approach that is proposed in [51]. The results are demonstrated in Fig. 5.18. Notice that, for both approaches, the energy consumption is inversely proportional to the PO value. Hence, the longer the schedule period, the lower the energy consumption. However, the TWBS modifies the original superframe structure of IEEE 802.15.4 in order to support two-way communications by enabling each cluster to be active twice within the schedule period. Hence, compared to our approach, each cluster-head is required to transmit two beacon frames within each schedule period which leads to additional energy consumption as shown in Fig. 5.18.

5.10.6 Network Reliability

The reliability of the network, as a function to the loss rate and the number of re-transmissions, is calculated as the percentage of the successful transmissions for each data flow from the source node to the sink node. In Fig. 5.19, we illustrate the reliability of the network presented in Fig. 5.1 based on the schedule obtained by DTDMA$^{\mathrm{mcd}}$ and the schedule obtained TWBS. The results show, for both algorithms, that the higher the number of re-transmissions and the lower the loss rate, the more the packets that are dispatched by the source nodes reach the sink nodes. Hence, the overall reliability of the network is increased. However, since the TWBS requires that each cluster is active twice within the schedule period, and consequently, each cluster-head transmits two beacon frames each schedule period to its child nodes to keep synchronization, then compared to our approach, the reliability of the network decreases in the case either beacon is not received successfully.

5.11 Conclusion

In this chapter, we aimed to further support to the on-the-fly deployment and configuration QoS property by focusing on the distributed methodologies. In particular, we considered a realistic model where the nodes are assumed to be self-organized into Cluster-Tree topology with a single-collision domain or multiple-collision domains. The data flows within the network are time-bounded and might traverse the network with opposite directions to each other. To enable each cluster within the network to efficiently configure all the required parameters for the allocation of the time-slots within the schedule period, we proposed exact and heuristic distributed TDMA algorithms. The exact algorithm targets the case of single-collision domain while the heuristic algorithm targets the case of the multiple-collision domains since it applies the spatial reuse of the transmission medium. The proposed algorithms are based on the graph theory such as the distributed shortest path, the distributed topological ordering, and the distributed graph coloring algorithms.

We proved, by the experimental results and the simulation scenarios, that the algorithms well-suit the scarce resources of the sensor nodes. In particular, the overhead of the algorithms, in term of the elapsed time up to the time instant at which the schedule is configured, is small even for networks with thousands of nodes. Consequently, the energy consumption is law. We also demonstrated the energy consumption of the nodes while running the heuristic algorithm as a function to the number of the loss rate of the channel, and the average number of competitors within the network.

The results confirm that the energy consumption directly proportional with the loss rate and the average number of competitors. This is due to the increase in the number of transmitted packets in either case.

Also, we discussed the energy consumption and the network reliability due to the obtained TDMA cluster schedule in comparison with TWBS approach. The energy consumption is demonstrated as a function to the length of the schedule period while the network reliability is demonstrated as a function to the loss rate of the channel. The results confirm that our approach leads to more energy efficient and reliable network.

Chapter 6

Conclusion

The rapid utilization of Wireless Sensor Networks (WSNs) by modern networked embedded systems is the primary motivation that has driven the work presented in this thesis. In that direction, the thesis aimed to further develop the support for the wireless infrastructure for the Internet of Things (IoT) and Industry 4.0 paradigms. In particular, the modern networked embedded applications, such as industrial monitoring and control applications, tend to connect a tremendous number of small and smart devices to monitor and control everything, everywhere even in hard to reach and hazardous environments. However, due to the scarce resources of the sensor nodes (e.g., memory size, processor power, and battery capacity) and due to the specific requirements for the target application, it is crucial to keep in mind a particular set of Quality of Service (QoS) properties while developing new concepts and solution concerning WSNs. For example, collision avoidance, energy efficiency, timeliness, and network reliability are of paramount importance to be considered. Other requirements, such as on-the-fly deployment and configuration, are essential.

The IEEE 802.15.4/ZigBee standards are leading technologies for low-cost, low-power, and low-rate WSNs. Besides, IEEE 802.15.4/ZigBee Cluster-Tree topology, with beacon-enabled mode, is one of the infrastructure-based WSNs technology that is popular for performance guarantee. However, the current state-of-the-art reveals a strong immatureness and an evident lack of solutions concerning the above mentioned stringent required QoS properties. In that direction, the thesis push forward the support to the technology through the design of collision-free TDMA cluster scheduling algorithms while considering a realistic system model that relies as much as possible upon to the real application scenarios. Therefore, this thesis considers a beacon-enabled IEEE 802.15.4/ZigBee Cluster-Tree WSN topology. The traffic within the network is organized into a set of data flows with a given set of parameters such as source nodes, sink node, and end-to-end deadline. The traffic goes in both direction in order to fulfill the necessity of the support to the industrial monitoring and control applications where the sensed data and the control data go in opposite directions (i.e., from/to the field devices).

The proposed TDMA cluster scheduling algorithms provide optimized and novel techniques/methods in order to improve the above mentioned QoS properties. In particular, the collision avoidance and the timeliness QoS properties are addressed through the proper allocation and the proper ordering of the time-slots that are allocated to each cluster. The energy efficiency of the schedule is accomplished by adjusting the duty cycle of each cluster. In other words, by allocating the minimal number of the time-slots to each node, based on the amount of data to be transmitted by the node, and by maximizing the duration at which each node remains in power-saving mode. Such an approach leads to the prolongation of the lifetime of the network. However, since timeliness is inconsistent with energy efficiency, then a fair trade-off between energy efficiency and timeliness is required. The communication reliability is accomplished by the re-transmission and acknowledgment mechanism that is naturally supported by IEEE 802.15.4/ZigBee standards. Since the network reliability is inconsistent with timeliness and energy efficiency, then a fair trade-off is also required. In fact, the number of re-transmissions must be bounded for proper analysis and proper functionality of the proposed algorithms. To support on-the-fly deployment and configuration QoS property, distributed methods are proposed in order to enable each cluster within the network to come up with its allocated portion within the schedule and to configure all the required parameters. The distributed methods also guarantee collision avoidance, energy efficiency, timeliness, and network reliability QoS properties.

The complexity of the TDMA cluster scheduling problem while considering multiple-collision domains Cluster-Tree topology is $\mathcal{NP}$-hard. Hence, Chapter 3 simplifies the problem by assuming single-collision domain Cluster-Tree topology (i.e. at most, one cluster can be active at any given time) and by expressing the precise end-to-end deadline, given in time units, into the maximum number of crossed periods. Both simplifications lead to polynomial time complexity instead of $\mathcal{NP}$-hard. Furthermore, it proposes TDMA$^{\text{scd}}$ as an exact, fast and light algorithm that ensures collision avoidance, energy efficiency, timeliness, and network reliability QoS properties. The TDMA$^{\text{scd}}$ algorithm is based on graph theory algorithms, such as shortest path algorithm and topological ordering, and can solve large-size instances in a short time. Since the simulation is an essential approach to developing and evaluating the systems, simulation scenarios are accomplished in Opnet Modeler 17.5 to demonstrate the impact of the number of re-transmissions on the network reliability, energy efficiency and timeliness of the data flows. Also, the energy consumption and the timeliness of the data transmissions as a function to the duty cycle of the nodes are demonstrated. The results show that a fair trade-off between energy efficiency, timeliness and network reliability is required with respect to the target application.

The Chapter 4 extends and complete the work presented in Chapter 3 by considering a realistic system model with multiple-collision domains Cluster-Tree topology WSNs. Hence, the spatial reuse of the transmission medium is considered for better bandwidth utilization. The chapter proposes E_TDMA^{mcd} as an exact cluster scheduling algorithm that is based on ILP. The E_TDMA^{mcd} algorithm addresses collision avoidance, energy efficiency, timeliness and network reliability QoS properties for small-size instances. To cope with the complexity of the ILP, we implement H_TDMA^{mcd} as a heuristic scheduling algorithm based on graph theory and combinatorial optimization problems such as shortest path and graph coloring algorithms. The H_TDMA^{mcd} algorithm addresses collision avoidance, energy efficiency, timeliness and network reliability QoS properties for large-size instances with thousands of nodes. Furthermore, we demonstrate, through the benchmarks and the comparison with TDCS algorithm [25], that the heuristic algorithm is efficient in both computational time and solution quality. Moreover, the simulation model, accomplished in Opnet Modeler 17.5, reveals as expected that a fair trade-off is required between energy efficiency, timeliness and network reliability QoS properties.

In Chapter 5, we further support the on-the-fly deployment and configuration. In particular, we consider a realistic model with single-collision domain and multiple-collision domains Cluster-Tree topology. Furthermore, we propose DTDMA^{scd} as an exact distributed scheduling algorithm for single-collision domain Cluster-Tree topology and DTDMA^{mcd} as a heuristic distributed scheduling algorithm for multiple-collision domains Cluster-Tree topology. In contrast to DTDMA^{scd}, the DTDMA^{mcd} supports the spatial reuse of the transmission medium for better bandwidth utilization and consequently for better support of large-scale networks. The algorithms are based on distributed graph theory algorithms such as distributed shortest path, distributed topological ordering and distributed graph coloring algorithms. In particular, the algorithms enable each node within the network to come up with its allocated time-slots and configure all other parameters, such as the number of re-transmissions and the duration of the power-saving mode, in order to meet all the specified QoS properties (i.e., collision avoidance, energy efficiency, timeliness, and network reliability QoS properties). The experimental results, the comparison with existing works, and the simulation scenarios prove that the overhead of the algorithms, in term of the elapsed time up to the time instant at which the schedule is configured and in terms of energy consumption, is small even for networks with thousands of nodes.

Bibliography

[1] Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), 2006.

[2] IEEE P802.15 Wireless Personal Area Networks: Proposal for Factory Automation, 2009.

[3] Aasem Ahmad and Zdeněk Hanzálek. Distributed Real Time TDMA Scheduling Algorithm for Tree Topology WSNs. *IFAC-PapersOnLine*, 50(1): 5926–5933, 2017. ISSN 2405-8963. `http://www.sciencedirect.com/science/article/pii/S240589631732061X`. 20th IFAC World Congress.

[4] Aasem Ahmad and Zdeněk Hanzálek. Distributed Real Time TDMA Schedule for ZigBee-Like Cluster-Tree Topology. *ACM Transaction of Sensor Networks*, 2019 **(major revision)**.

[5] Mohammad M. Abdellatif, Jose Manuel Oliveira, and Manuel Ricardo. The Self-Configuration of Nodes Using RSSI in a Dense Wireless Sensor Network. *Telecommun Syst*, 2016. `https://doi.org/10.1007/s11235-015-0105-7`.

[6] A. Ahmad and Z. Hanzálek. ZigBee Cluster Tree Formation for Time-Bounded Data Flows in One Collision Domain. In *2015 IEEE World Conference on Factory Communication Systems (WFCS)*, pages 1–4, May 2015. DOI: 10.1109/WFCS.2015.7160572.

[7] A. Ahmad and Z. Hanzálek. An Energy Efficient Schedule for IEEE 802.15.4/ZigBee Cluster Tree WSN with Multiple Collision Domains and Period Crossing Constraint. *IEEE Transactions on Industrial Informatics*, 14(1): 12–23, Jan 2018. ISSN 1551-3203. DOI: 10.1109/TII.2017.2725907.

[8] A. Ahmad, Z. Hanzálek, and C. Hanen. A Polynomial Scheduling Algorithm for IEEE 802.15.4/ZigBee Cluster Tree WSN with One Collision Domain and Period Crossing Constraint. In *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*, pages 1–8, Sept 2014.

[9] ZigBee Alliance. ZigBee Specification (Document 053474r20); ZigBee Alliance: San Ramon, CA, USA, 2012.

[10] G. Anastasi, M. Conti, and M. Di Francesco. A Comprehensive Analysis of the MAC Unreliability Problem in IEEE 802.15.4 Wireless Sensor Networks. *IEEE Trans. on Industrial Informatics*, 7(1), Feb 2011. ISSN 1551-3203.

[11] A. Bhatia and R. C. Hansdah. A Distributed TDMA Slot Scheduling Algorithm for Spatially Correlated Contention in WSNs. *Mobile Information Systems*, 2015, 2015.

[12] O. Bonaventure. *Computer Networking : Principles, Protocols and Practice*. The Saylor Foundation, 2011.

[13] H. Boujelben, O. Gaddour, and M. Abid. Enhancement and Performance Evaluation of a Multicast Routing Mechanism in ZigBee Cluster-Tree Wireless Sensor Networks. In *2013 10th International Multi-Conference on Systems, Signals and Devices (SSD)*, pages 1–8, March 2013.

[14] Simone Brienza, Manuel Roveri, Domenico De Guglielmo, and Giuseppe Anastasi. Just-in-Time Adaptive Algorithm for Optimal Parameter Setting in 802.15.4 WSNs. *ACM Trans. Auton. Adapt. Syst.*, 10(4): 27:1–27:26, January 2016. ISSN 1556-4665.

[15] C. Caione, D. Brunelli, and L. Benini. Distributed Compressive Sampling for Lifetime Optimization in Dense Wireless Sensor Networks. *IEEE Trans. on Industrial Informatics*, 8(1), Feb 2012. ISSN 1551-3203.

[16] N. Choudhury, R. Matam, M. Mukherjee, and L. Shu. Beacon Synchronization and Duty-Cycling in IEEE 802.15.4 Cluster-Tree Networks: A Review. *IEEE Internet of Things Journal*, 5(3): 1765–1788, June 2018. ISSN 2327-4662. DOI: 10.1109/JIOT.2018.2827946.

[17] R. Diestel. *Graph Theory*. Springer-Verlag, 2005.

[18] Yuemin Ding and Seung Ho Hong. CFP Scheduling for Real-Time Service and Energy Efficiency in the Industrial Applications of IEEE 802.15.4. *Communications and Networks, Journal of*, 15(1): 87–101, Feb 2013. ISSN 1229-2370.

[19] F. Dobslaw, T. Zhang, and M. Gidlund. End-to-End Reliability-Aware Scheduling for Wireless Sensor Networks. *IEEE Transactions on Industrial Informatics*, 12(2): 758–767, April 2016. ISSN 1551-3203. DOI: 10.1109/TII.2014.2382335.

[20] K. Erciyesl. *Distributed Graph Algorithms for Computer Networks.* Springer-Verlag, 2005.

[21] S. C. Ergen and P. Varaiya. PEDAMACS: Power Efficient and Delay Aware Medium Access Protocol for Sensor Networks. *IEEE Transactions on Mobile Computing*, 5(7): 920–930, July 2006. ISSN 1536-1233.

[22] Sinem Coleri Ergen and Pravin Varaiya. TDMA Scheduling Algorithms for Wireless Sensor Networks. *Wirel. Netw.*, 16(4), May 2010. ISSN 1022-0038.

[23] Hamid Reza Faragardi, Maryam Vahabi, Hossein Fotouhi, Thomas Nolte, and Thomas Fahringer. An Efficient Placement of Sinks and SDN Controller Nodes for Optimizing the Design Cost of Industrial IoT Systems. *Special Issue Meta-heuristics in Cloud Computing*, 47: 1–27, June 2018. `http://www.es.mdh.se/publications/5159-`.

[24] G. Franchino, G. Buttazzo, and M. Marinoni. Bandwidth Optimization and Energy Management in Real-Time Wireless Networks. *ACM Trans. Embed. Comput. Syst.*, 15(3): 41:1–41:29, March 2016. ISSN 1539-9087.

[25] Z. Hanzálek and P. Jurčík. Energy Efficient Scheduling for Cluster-Tree Wireless Sensor Networks with Time-Bounded Data Flows: Application to IEEE 802.15.4/ZigBee. *IEEE Transaction on Industrial Informatics*, 6(3), August 2010.

[26] Hanzalek, Zdenek and Hanen, Claire. The Impact of Core Precedences in a Cyclic RCPSP with Precedence Delays. *Journal of Scheduling*, 18 (3): 275–284, 2015. ISSN 1099-1425.

[27] Seung Ho Hong. Scheduling Algorithm of Data Sampling Times in the Integrated Communication and Control Systems. *Control Systems Technology, IEEE Transactions on*, 3(2): 225–230, Jun 1995. ISSN 1063-6536.

[28] P. Huang, L. Xiao, S. Soltani, M. W. Mutka, and N. Xi. The Evolution of MAC Protocols in Wireless Sensor Networks: A Survey. *IEEE Communications Surveys Tutorials*, 15(1): 101–120, First 2013. ISSN 1553-877X.

[29] P. Jurčík and Z. Hanzálek. Simulation Study of Energy Efficient Scheduling for IEEE 802.15.4/ZigBee Cluster-Tree Wireless Sensor Networks with Time-Bounded Data Flows. In *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept 2010.

[30] Jin-Woo Kim and Jae-Wan Kim. Energy Efficient Clustering Algorithm for the Mobility Support in an IEEE 802.15.4 Based Wireless Sensor Network. *Wireless Networks*, 2019. ISSN 1572-8196. DOI: 10.1007/s11276-019-01939-2.

[31] Taehong Kim, Seong Hoon Kim, and Daeyoung Kim. Distributed Topology Construction in ZigBee Wireless Networks. *Wireless Personal Communications*, 2018. ISSN 1572-834X. DOI: 10.1007/s11277-018-5905-0.

[32] A. Koubâa, A. Cunha, M. Alves, and E. Tovar. TDBS: A Time Division Beacon Scheduling Mechanism for ZigBee Cluster-Tree Wireless Sensor Networks. *Real-Time Systems Journal*, 40(3): 321–354, October 2008.

[33] S. Kumar and H. Kim. Energy Efficient Scheduling in Wireless Sensor Networks for Periodic Data Gathering. *IEEE Access*, 7: 11410–11426, 2019. ISSN 2169-3536. DOI: 10.1109/ACCESS.2019.2891944.

[34] Jae-Hyoung Lee and Sung Ho Cho. Tree TDMA MAC Algorithm Using Time and Frequency Slot Allocations in Tree-Based WSNs. *Wireless Personal Communications*, pages 1–23, 2017. ISSN 1572-834X.

[35] Erico Leão, Carlos Montez, Ricardo Moraes, Paulo Portugal, and Francisco Vasques. Superframe Duration Allocation Schemes to Improve the Throughput of Cluster-Tree Wireless Sensor Networks. *Sensors*, 17(2), 2017. ISSN 1424-8220. DOI: 10.3390/s17020249.

[36] Erico Leão, Carlos Montez, Ricardo Moraes, Paulo Portugal, and Francisco Vasques. Alternative Path Communication in Wide-Scale Cluster-Tree Wireless Sensor Networks Using Inactive Periods. *Sensors*, 2017. ISSN 1424-8220. DOI: 10.3390/s17051049.

[37] Wenjuan Liu, Dongmei Zhao, and Gang Zhu. End-to-End Delay and Packet Drop Rate Performance for a Wireless Sensor Network with a Cluster-Tree Topology. *Wireless Communications and Mobile Computing*, 14(7), 2014. ISSN 1530-8677.

[38] J. Long, M. Dong, K. Ota, and A. Liu. A Green TDMA Scheduling Algorithm for Prolonging Lifetime in Wireless Sensor Networks. *IEEE Systems Journal*, 2015.

[39] K. Moriyama and Y. Zhang. An Efficient Distributed TDMA MAC Protocol for Large-Scale and High-Data-Rate Wireless Sensor Networks. In *IEEE 29th International Conference on Advanced Information Networking and Applications (AINA)*, March 2015.

[40] L. Palopoli, R. Passerone, and T. Rizano. Scalable Offline Optimization of Industrial Wireless Sensor Networks. *IEEE Trans. on Industrial Informatics*, 7(2), May 2011. ISSN 1551-3203.

[41] C. E. Perkins and E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*, Feb 1999. DOI: 10.1109/MCSA.1999.749281.

[42] R. Severino, N. Pereira, and E. Tovar. Dynamic Cluster Scheduling for Cluster-Tree WSNs. *SpringerPlus Commun. Netw*, pages 1–17, 2014.

[43] Ricardo Severino, Sana Ullah, and Eduardo Tovar. A cross-layer qos management framework for zigbee cluster-tree networks. *Telecommunication Systems*, Nov 2016. ISSN 1572-9451. DOI: 10.1007/s11235-015-0128-0.

[44] Shahin Farahani. *ZigBee Wireless Networks and Transceivers*. Newnes is an imprint of Elsevier, Newton, MA, 2008.

[45] W. Shen, T. Zhang, F. Barac, and M. Gidlund. PriorityMAC: A Priority-Enhanced MAC Protocol for Critical Traffic in Industrial Wireless Sensor and Actuator Networks. *IEEE Trans. on Industrial Informatics*, 10(1), Feb 2014. ISSN 1551-3203.

[46] Ivan Tomasic, Nikola Petrovic, Hossein Fotouhi, Mats Björkman, and Maria Lindén. IoT Enabled Monitoring of Patients' Environmental Parameters Supported by OpenWSN, OpenMote, and Relational Databases. In *Medicinteknikdagarna 2018*, October 2018. `http://www.es.mdh.se/publications/5295-`.

[47] E. Toscano and L. L. Bello. Multichannel Superframe Scheduling for IEEE 802.15.4 Industrial Wireless Sensor Networks. *IEEE Trans. on Industrial Informatics*, 8(2): 337–350, May 2012. ISSN 1551-3203.

[48] A. Willig. Recent and Emerging Topics in Wireless Industrial Communications: A Selection. *IEEE Trans. on Industrial Informatics*, 4(2): 102–124, May 2008.

[49] Y. Wu, K. S. Liu, J. A. Stankovic, T. He, and S. Lin. Efficient Multichannel Communications in Wireless Sensor Networks. *ACM Trans. Sen. Netw.*, 2016.

[50] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proc. of the 21st Annual Joint*

Conf. of the IEEE Computer and Communications Societies, June 2002.

[51] L. Yeh and M. Pan. Beacon Scheduling for Broadcast and Convergecast in ZigBee Wireless Sensor Networks. *Computer Communications*, 2014.

[52] Seong-Eun Yoo, Poh Kit Chong, Daeyoung Kim, Yoonmee Doh, Minh-Long Pham, Eunchang Choi, and Jaedoo Huh. Guaranteeing Real-Time Services for Industrial Wireless Sensor Networks With IEEE 802.15.4. *Industrial Electronics, IEEE Transactions on*, 57(11): 3868–3876, Nov 2010. ISSN 0278-0046.

[53] B. Yu, J. Li, and Y. Li. Distributed Data Aggregation Scheduling in Wireless Sensor Networks. In *INFOCOM 2009, IEEE*, pages 2159–2167, April 2009.

[54] M. M. Zanjireh and H. Larijani. A Survey on Centralised and Distributed Clustering Routing Algorithms for WSNs. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–6, May 2015. DOI: 10.1109/VTCSpring.2015.7145650.

[55] T. Zheng, M. Gidlund, and J. Åkerberg. Medium Access Protocol Design for Time-Critical Applications in Wireless Sensor Networks. In *Factory Communication Systems (WFCS), 2014 10th IEEE Workshop on*, pages 1–7, May 2014.

Appendix A

Curriculum Vitae

Aasem Ahmad was born in Latakia, Syria, in 1986. He received his degree in electronic engineering with honor from Tishreen university, Latakia, Syria, in 2009. His thesis focused on modeling and simulation of heterogeneous systems. He was a Teacher Assistant in the Department of Computer Engineering and Automated Control, Tishreen University, from 2010 to 2013. His teaching activities cover object oriented programming (C++), data base programming and pattern recognitions.

Since 2013, he has been working towards the doctor of philosophy degree (Ph.D.) in robotics and control engineering at the Czech Technical University in Prague, on the topic of optimization scheduling algorithms for wireless sensor networks. His teaching activities cover the subject of combinatorial optimization where he also participated in the preparation of the educational material.

The research results of Aasem Ahmad were presented in several international conferences (e.g., ETFA 2014, WFCS 2015, IFAC 2017) and in impacted international journals: IEEE Transaction of Industrial Informatics and ACM Transaction of Sensor networks (major revision). Furthermore, He reviewed several conferences and journal papers.

Aasem Ahmad
Prague, May 2019

Appendix B

List of Author's Publications

List of publications and technical reports related to this thesis is included in this appendix.

Publications in Journals with Impact Factor

Aasem Ahmad and Zdeněk Hanzálek, An Energy Efficient Schedule for IEEE 802.15.4/ZigBee Cluster Tree WSN with Multiple Collision Domains and Period Crossing Constraint, *in IEEE Transactions on Industrial Informatics*, vol. 14, no. 1, pp. 12-23, Jan. 2018. DOI: `10.1109/TII.2017.2725907`. **Co-authorship 50 %, Journal rank Q1**.

Aasem Ahmad and Zdeněk Hanzálek, Distributed Real Time TDMA Schedule for ZigBee-Like Cluster-Tree Topology, *ACM Transaction of Sensor Networks*, **(major revision), Co-authorship 50 %, Journal rank Q2**.

International Conferences and Workshops

Aasem Ahmad, Zdeněk Hanzálek and Clair Hanen, A Polynomial Scheduling Algorithm for IEEE 802.15.4/ZigBee Cluster-Tree WSN with One Collision Domain and Period Crossing Constraint, *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), Barcelona, 2014*, pp. 1-8. DOI: `10.1109/ETFA.2014.7005182`. **Co-authorship 33.3 %**.

Aasem Ahmad and Zdeněk Hanzálek, ZigBee Cluster-Tree Formation for Time-Bounded Data Flows in One Collision Domain, *2015 IEEE World Conference on Factory Communication Systems (WFCS), Palma de Mallorca*, 2015, pp. 1-4. DOI: `10.1109/WFCS.2015.7160572`. **Co-authorship 50 %**.

Aasem Ahmad and Zdeněk Hanzálek, Distributed Real Time TDMA Scheduling Algorithm for Tree Topology WSNs. IFACPapersOnLine, 50(1): 5926–5933, 2017. ISSN 2405-8963. `http://www.sciencedirect.com/science/article/pii/S240589631732061X`. 20th IFAC World Congress. **Co-authorship 50 %. Presented in IFAC 2017 and published in IFACPapersOnLine impacted Journal - Journal rank Q3**.

Other Publications

Aasem Ahmad, Energy Efficient Cluster Schedule for Time Bounded Data Flows in One and Multiple Collision Domains ZigBee Cluster-Tree WSNs, *Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic*, 2015. **Co-authorship 100 %**.

Aasem Ahmad
Prague, May 2019

This thesis is aimed to further support wireless infrastructure for the Internet of Things (IoT) and industry 4.0 paradigms. Its main contributions are as follows:

1. Realistic system model with single-collision domain and multiple-collision domains Cluster-Tree topology WSNs and time-bounded multi-hops data transmissions.

2. Design and implement exact and heuristic, centralized and distributed TDMA scheduling algorithms that provide a set of QoS such as collision avoidance, energy efficiency, timeliness, network scalability and reliability, and on-the-fly deployment and configuration.

3. Verify the proposed algorithms on benchmark instances and compare them with the existing works.

4. Simulation scenarios are accomplished using Opnet Modeler 17.5 to further verify the proposed solutions.