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

in collaboration with

CISTER-ISEP Research Unit
Polytechnic Institute of Porto
Portugal

REAL-TIME COMMUNICATION OVER CLUSTER-TREE WIRELESS SENSOR NETWORKS

a doctoral thesis submitted in partial fulfilment of
the requirements for the degree of

Doctor of Philosophy (Ph.D.)

by

Petr Jurčík

Prague, January 2010

Ph.D. Programme: *Electrical Engineering and Information Technology*
Branch of study: *Control Engineering and Robotics*

Supervisor: *Doc. Dr. Ing. Zdeněk Hanzálek*
Co-supervisor: *Anis Koubâa, Ph.D.*

© Copyright by
Petr Jurčík
All rights reserved
January 2010

To me and my family.

Acknowledgements

First of all, I would like to express my thanks to my thesis supervisor, Zdeněk Hanzálek, and thesis co-supervisors, Anis Koubâa and Mário Alves, for their patient supervision, help and support leading towards this thesis. I am also grateful to Eduardo Tovar, Ricardo Severino and all other people, that I met during my stay at CISTER-ISEP Research Unit in Porto, for the opportunity to be part of their group, for their help and especially for creating an excellent working environment. Finally, I wish to express my appreciation for numerous conference and journal reviewers provided me with many useful comments and valuable feedbacks on all submitted papers. Last, but not least, acknowledgements belong to my family, friends and Řadoff.

This work was carried out in collaboration with CISTER-ISEP Research Unit, and it was funded by FCT under the CISTER Research Unit (FCT UI 608), by the PLURALITY project (CONCREEQ/900/2001), by the ARTIST2 NoE and by the Ministry of Industry and Trade of the Czech Republic under the project 61 03001.

Czech Technical University in Prague *Petr Jurčík*
January 2010

Abstract

Modelling and simulation of the fundamental performance limits of Wireless Sensor Networks (WSNs) is of paramount importance to understand their behaviour under the worst-case conditions and to make the appropriate design choices. This is particular relevant for time-sensitive WSN applications, where the timing behaviour of the network protocols impacts on the correct operation of these applications. Furthermore, energy efficiency is a key requirement to be fulfilled in these applications since the wireless nodes are usually battery-powered. In that direction this thesis contributes with an accurate simulation model of the IEEE 802.15.4/ZigBee protocols and an analytical methodology for the worst-case analysis and dimensioning of a static or even dynamically changing cluster-tree WSN where the data sink can either be static or mobile. The thesis is focused on the study of WSNs with cluster-tree topology because it supports predictable and energy efficient behaviour, which is suited for time-sensitive applications using battery-powered nodes. On the other side, in contrast with the star and mesh topologies, the cluster-tree topology expresses several challenging and open research issues such as a precise cluster scheduling to avoid inter-cluster collisions (messages/beacons transmitted from nodes in different overlapping clusters). Hence, the next objective is to find the collision-free periodic schedule of clusters' active portions, called Time Division Cluster Schedule (TDCS), while minimizing the energy consumption of the nodes and meeting all data flows' parameters. The thesis also shows how to apply the proposed methodologies to the specific case of IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs, as an illustrative example that confirms the applicability of general approach for specific protocols. Finally, the validity and accuracy of the simulation model and methodologies are demonstrated through the comprehensive experimental and simulation studies. Using the proposed analytical methodologies and simulation model, system designers are able to easily configure the IEEE 802.15.4/ZigBee cluster-tree WSN for a given application-specific Quality of Service (QoS) requirements prior to the network deployment.

Keywords: cluster-tree; energy efficiency; IEEE 802.15.4; Network Calculus; quality of service; real-time; simulation; wireless sensor network; ZigBee.

Goals and objectives

The main goals of this work have been set as follows.

1. The design, implementation and evaluation of an accurate simulation model for IEEE 802.15.4 and ZigBee protocols focusing on the Guaranteed Time Slot (GTS) mechanism and ZigBee hierarchical routing strategy in beacon-enabled cluster-tree Wireless Sensor Networks (WSNs).
2. The formulation, implementation and evaluation of a methodology that solves the energy efficient clusters' scheduling problem in a static cluster-tree WSN with a predefined set of time-bounded data flows, assuming bounded communication errors.
3. The analysis of the interdependence among the reliability of data transmission, the energy consumption of the nodes and the timeliness in IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs.
4. The formulation, implementation and evaluation of a methodology that enables quick and efficient worst-case dimensioning of network resources in a static or even dynamically changing cluster-tree WSN where a static or mobile sink gathers data from all sensor nodes, assuming bounded communication errors.

Contents

1 Introduction 1
1.1 Outline and Contribution 3

2 Overview of IEEE 802.15.4 and ZigBee 7
2.1 Introduction 7
2.2 Physical layer 8
2.3 Data link layer 9
2.4 ZigBee network layer 12

3 IEEE 802.15.4/ZigBee simulation model 17
3.1 Introduction 17
3.2 Related work 18
3.3 Simulation model 20
  3.3.1 Simulation model structure 20
  3.3.2 User-defined attributes 22
3.4 Simulation setup 24
  3.4.1 Simulation vs. analytical models 24
3.5 Performance evaluation 25
  3.5.1 Impact of SO on the GTS throughput 25
  3.5.2 Impact of SO on the media access delay 31
3.6 Conclusions 35

4 Energy efficient scheduling for cluster-tree WSNs 37
4.1 Introduction 37
4.2 Related work 39
4.3 System model 41
  4.3.1 Cluster-tree topology model 41
  4.3.2 Data flow model 44
  4.3.3 Cyclic nature 45
  4.3.4 Collision domains 47

4.4 Time Division Cluster Scheduling 48
4.4.1 Duration of the cluster's active portion 49
4.4.2 TDCS formulated as a cyclic extension of RCPS/TC . 51
4.4.3 Graph of the communication tasks 54
4.4.4 Solution of the scheduling problem by ILP algorithm . 56
4.5 Time complexity 58
4.6 Simulation study 61
4.6.1 Simulation setup 61
4.6.2 Simulation results 63
4.7 Conclusions 65
4.A Table of symbols 68

5 Dimensioning and worst-case analysis of cluster-tree WSNs 69
5.1 Introduction 69
5.2 Related work 71
5.3 Background on Network Calculus 74
5.4 System model 79
5.4.1 Cluster-tree topology model 79
5.4.2 Data flow model 80
5.4.3 Time Division Cluster Schedule 83
5.5 Data flow analysis 85
5.5.1 Upstream direction 85
5.5.2 Downstream direction 88
5.6 Worst-case network dimensioning 92
5.6.1 Per-router resources analysis 92
5.6.2 End-to-end delay analysis 95
5.7 Application to IEEE 802.15.4/ZigBee 98
5.7.1 Guaranteed bandwidth of a GTS 98
5.7.2 Characterization of the service curve 100
5.7.3 IEEE 802.15.4/ZigBee cluster-tree WSN setup 102
5.8 Performance evaluation 104
5.8.1 Network setup 105
5.8.2 Analytical evaluation 107
5.8.3 Experimental evaluation 110
5.9 Discussion on mobility support 116
5.9.1 Adapting the logical topology 116
5.9.2 Routing protocol 117
5.10 Conclusions 120
5.A Table of symbols 121

6 Conclusions 123

Bibliography **132**

Index **134**

Curriculum vitae **134**

Author's publications **135**

List of Figures

2.1 Structure of the IEEE 802.15.4/ZigBee frames. 9
2.2 Superframe structure of IEEE 802.15.4. 10
2.3 The Inter-Frame Spacing. 12
2.4 IEEE 802.15.4/ZigBee network topologies. 13

3.1 The structure of the simulation model. 21
3.2 The user-defined attributes of the GTS mechanism. 23
3.3 The utilization of the GTS bandwidth. 26
3.4 GTS throughput as a function of the rate. 27
3.5 GTS throughput: simulation vs. analytical model. 28
3.6 GTS throughput as a function of the buffer capacity. 29
3.7 GTS throughput: simulation vs. analytical model. 30
3.8 The media access delay as a function of the arrival data rate. 32
3.9 The media access delay as a function of the buffer capacity. . 33
3.10 Average vs. maximum media access delay. 34
3.11 Delay of the frames stored in the FIFO buffer. 35
3.12 The media access delay: simulation vs. analytical model. . . . 36

4.1 Cluster-tree topology with 2 time-bounded data flows. 42
4.2 Timing among clusters 1,2 and 6 from Figure 4.1. 43
4.3 Schedules for data flows in Figure 4.1. 46
4.4 The carrier-sense area and collision domain. 47
4.5 In-trees of dummy-tasks relating to the data flows. 53
4.6 Graph G of the tasks corresponding to example in Figure 4.1. 55
4.7 ILP formulation for cyclic extension of the scheduling problem. 56
4.8 Gantt chart of TDCS including flows 1 and 2. 57
4.9 Time line of TDCS corresponding to the example in Figure 4.1. 59
4.10 The simulation scenario in Opnet Modeler. 62
4.11 Reliability and sum of energy consumption. 63
4.12 E2e delay and energy consumption as a function of BO. . . . 64

4.13 E2e delay as a function of $macMaxFrameRetries$. 65
4.14 E2e delay as a function of $macMaxFrameRetries$. 66

5.1 The basic system model in Network Calculus. 74
5.2 Example of a cumulative input and output functions. 75
5.3 The worst-case cluster-tree topology model. 81
5.4 General data flow model. 82
5.5 The queuing system model for the upstream direction. 86
5.6 The queuing system model for downstream direction. 89
5.7 The locations of a sink router and correspondent data flows. . 92
5.8 Downstream bandwidth increase factor. 94
5.9 The worst-case service latencies. 101
5.10 The test-bed deployment and sensory data traffic. 105
5.11 The GUI of the Matlab analytical model. 106
5.12 The worst-case delay and bandwidth. 108
5.13 The worst-case delay and buffer requirement. 109
5.14 The worst-case buffer requirement. 111
5.15 Theoretical vs. experimental data traffic. 112
5.16 The theoretical vs. experimental delay bounds. 114
5.17 The worst-case and maximum end-to-end delays. 115
5.18 Two approaches to mobile sink behaviour. 117
5.19 The route update algorithm. 119

List of Tables

2.1 Star vs. mesh vs. cluster-tree topologies. 15

3.1 Relation between data rate and Packet Interarrival Time. . . 27

4.1 The user-defined parameters of the data flows from Figure 4.1. 44
4.2 Guaranteed bandwidth of clusters' superframes. 51
4.3 Time complexity of TDCS algorithm. 60
4.4 The user-defined parameters. 63
4.5 Table of symbols. 68

5.1 Buffer requirements: theoretical vs. experimental results. . . 113
5.2 Delay bounds: theoretical vs. experimental results. 114
5.3 Table of symbols. 122

List of Acronyms

BI	Beacon Interval
BO	Beacon Order
CAP	Contention Access Period
CFP	Contention Free Period
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
GTS	Guaranteed Time Slot
FFD	Full-Function Device
IFS	Inter-frame Spacing
ILP	Integer Linear Programming
MAC	Medium Access Control
MPDU	MAC Protocol Data Unit
MSDU	MAC Service Data Unit
NPDU	Network Protocol Data Unit
NSDU	Network Service Data Unit
PAN	Person Area Network
PPDU	Physical Protocol Data Unit
PSDU	Physical Service Data Unit
QoS	Quality of Service
RFD	Reduced-Function Device
SD	Superframe Duration
SO	Superframe Order
TDCS	Time Division Cluster Schedule
WC-TDCS	Worst-Case Time Division Cluster Schedule
WSN	Wireless Sensor Network
ZC	ZigBee Coordinator
ZED	ZigBee End Device
ZR	ZigBee Router

Chapter 1

Introduction

The tendency for the integration of computations with physical processes is pushing research on new paradigms for networked embedded systems design [1]. Wireless Sensor Networks (WSNs) have naturally emerged as enabling infrastructures for cyber-physical applications that closely interact with external stimulus. WSNs are mainly aimed at control and monitoring applications where relatively low data throughput and large scale deployment are the main system features. Furthermore, energy efficiency and timeliness are key requirements to be fulfilled in these applications since the wireless nodes are usually battery-powered and the end-to-end delays of time-sensitive messages must be bounded. For example, the emergency response system in a disaster area or intruder alarm system on the border line [2,3] both require time-bounded communications and long lifetime of entire network.

Wireless Sensor Networks may be installed and maintained for a fraction of the cost and time of an existing wired network. Wireless networks offer more flexibility and can provide sensing and actuating in previously hard-to-reach areas. In addition, WSNs may be installed in a hazardous or extreme environment where very specialized and costly procedures must be adhered. Since the wireless nodes are usually battery-powered, the network can be effectively used in environments where electricity is not available or some level of mobility is required (e.g. rotating parts of machines or linear position metering [4]). On the other side, using batteries requires effective power management.

Wireless Sensor Networks can be classified into two types, infrastructure-based networks and ad hoc (infrastructure-less) networks. The former is less flexible since it employs the pre-deployed and structured topology, but provides better support of predictable performance guarantees using deterministic routing protocols. Basically, the infrastructure-based networks rely on the use of contention-free MAC protocols (e.g. Time Division Multiple Access (TDMA) or token passing) to ensure collision-free and predictable access to the shared wireless medium, and the ability to perform end-to-end resource reservation. These represent important advantages of infrastructure-based networks when compared to what can be achieved in ad hoc networks, where contention-based MAC protocols and probabilistic routing protocols [5] are commonly used. The ad hoc network provides good flexibility to adaptive network changes, but at the cost of unpredictable performance. Hence, when predictable performance guarantees are the objective, it is suitable to rely on infrastructure-based WSNs such as cluster-tree. On the other side, the cluster-tree WSN expresses many challenging and open research issues in the area of real-time and energy efficient communications (e.g. a precise cluster scheduling to avoid inter-cluster collisions), which have been addressed in this thesis.

The WSN applications can be of many different types and can have different requirements [6]. For example, an environmental monitoring application that simply gathers temperature readings has less stringent requirements than a real-time tracking application using a set of wireless networked cameras. Therefore, it is crucial that sensor network resources are predicted in advance, to support the prospective applications with a predefined Quality of Service (QoS) such as end-to-end delay. Thus, it is important to have adequate methodologies to dimension network resources in a way that the requested QoS of the sensor network application is satisfied [7]. However, the provision of QoS has always been considered as very challenging due to the usually severe limitations of WSN nodes, such as the ones related to their energy, computational and communication capabilities, and due to communication errors resulting from the unreliable and time-varying characteristics of wireless channels [8]. Consequently, it is unrealistic to provide deterministic performance guarantees and support of hard real-time communications in a WSN. In general, no (wireless) communication channel is error-free thus being able to provide 100% guarantees.

Network communication protocols, e.g. at the data link layer, are able to detect most communication errors and, in some cases, correct some of them. The ultimate objective of communication protocols is to guarantee that messages arrive to the destination logically correct and on time. A corrupted or lost message can be detected by simple checksum or acknowledgement mechanisms, respectively, and it can be restored by a retransmission mechanism, for example. Note that all of these mechanisms are natively supported by the IEEE 802.15.4 standard [9]. However, each retransmission decreases throughput, increases the energy consumption of the nodes and the end-to-end communication delay such that a fair trade-off between reliability and timeliness of data transmission must be found. Even if the analysis has to deal with some unknown parameters, such as channel error, the maximum number of retransmissions must be bounded, otherwise, the analysis will not be possible. Using this bound, a system designer can perform capacity planning prior to network deployment to ensure the satisfaction of QoS requirements.

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 [9] and ZigBee specification [10], which are the leading communication technologies for low data rate, low cost and low power consumption WSNs. Chapter 3 presents an accurate IEEE 802.15.4/ZigBee simulation model and provides a novel methodology to tune the IEEE 802.15.4 parameters such that a better performance can be guaranteed. Assuming a static cluster-tree WSN with a set of multi-source mono-sink time-bounded data flows, the objective of Chapter 4 is to find the collision-free periodic schedule of clusters' active portions, called Time Division Cluster Schedule (TDCS), while minimizing the energy consumption of the nodes and meeting all data flows' parameters. Chapter 5 provides a simple yet efficient methodology based on Network Calculus for the worst-case dimensioning of static or even dynamically changing cluster-tree WSNs where the data sink can either be static or mobile and, consequently, the evaluation of the end-to-end delay bounds for time-sensitive data flows. Finally, the conclusions are drawn in Chapter 6.

1.1 Outline and Contribution

Three main parts can be identified within this thesis. The motivation that has driven the first part was the comprehensive performance evaluation of the real-time behaviour of the IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs. Thus, the Chapter 3 contributes an accurate Opnet simulation model of the IEEE 802.15.4/ZigBee protocols focusing on the implementation of the Guaranteed Time Slot (GTS) mechanism and ZigBee hierarchical routing strategy. The proposed simulation model is used to carry out a set of experiments and to compare the obtained simulation results with the ones that were previously obtained [11] using an analytical model based on Network Calculus. The behaviours of both models are roughly identical in terms of the GTS data throughput and the media access delay. Additionally, and probably more importantly, based on the simulation model a novel methodology is proposed to tune the IEEE 802.15.4 parameters such that a better performance can be guaranteed, both concerning maximizing the throughput of the allocated GTS as well as concerning minimizing media access delay.

In particular, the first part presents the following contributions:

1. An accurate simulation model of IEEE 802.15.4/ZigBee protocols that has been implemented in the Opnet network simulator.
2. A demonstration of the validity of proposed simulation model through an analytical model based on Network Calculus.
3. A novel methodology to tune the IEEE 802.15.4 parameters such that a better performance can be guaranteed.

The second part, relating to the Chapter 4, provides a clusters' scheduling mechanism, called Time Division Cluster Scheduling (TDCS), based on the cyclic extension of RCPS/TC (Resource Constrained Project Scheduling with Temporal Constraints) problem for a static cluster-tree WSN with a predefined set of time-bounded data flows, assuming bounded communication errors. The objective is to find a periodic schedule, which specifies when the clusters are active while avoiding possible inter-cluster collisions, meeting all end-to-end deadlines of time-bounded data flows and minimizing the energy consumption of the nodes by setting the TDCS period as long as possible. The performance evaluation of the TDCS scheduling tool shows that the problems with hundreds of nodes can be solved while using optimal solvers. The scheduling tool enables system designers to efficiently configure all the required parameters of the IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs in the network design time. The practical application of TDCS scheduling tool is demonstrated through the simulation study. In addition, using the simulation model the analysis in Section 4.6 shows how the maximum number of retransmissions impacts the reliability of data transmission, the energy consumption of the nodes and the end-to-end communication delay in the IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs.

In particular, the second part presents the following contributions:

1. A formulation of the clusters' scheduling problem by a cyclic extension of RCPS/TC. Using this formulation, the users are not restricted to a particular implementation but they can make a similar extension to any of the algorithms solving this type of problem.

2. A solution of cyclic extension of RCPS/TC by an Integer Linear Programming (ILP), where a grouping of Guaranteed Time Slots (GTS) leads to very efficient ILP model having a few decision variables.

3. An application of this methodology to a specific case of IEEE 802.15.4/ ZigBee beacon-enabled cluster-tree WSNs.

4. A time complexity evaluation of the proposed TDCS algorithm implemented in Matlab while using the simplex-based GLPK solver.

5. A simulation analysis of how the maximum number of retransmissions impacts the reliability of data transmission, the energy consumption of the nodes and the end-to-end communication delay.

The main outcome of the third part, relating to the Chapter 5, is the provision of a comprehensive methodology based on Network Calculus, which enables quick and efficient worst-case dimensioning of network resources (e.g. bandwidth and buffer size) in a static or even dynamically changing cluster-tree WSN where a static or mobile sink gathers data from all sensor nodes. Consequently, the worst-case performance bounds (e.g. end-to-end delay) can be evaluated for a cluster-tree WSN with bounded resources. This enables system designers to efficiently predict network resources that ensure a minimum QoS during extreme conditions (performance limits).

In particular, the third part presents the following contributions:

1. A formulation of a simple yet efficient methodology, based on Network Calculus, to characterize incoming and outgoing data traffic in each router in the cluster-tree WSN and to derive upper bounds on buffer requirements and per-hop and end-to-end delays for both upstream and downstream directions.

2. A description of how to instantiate this methodology in the design of IEEE 802.15.4/ZigBee cluster-tree WSNs, as an illustrative example that confirms the applicability of general approach for specific protocols.

3. A demonstration of the validity of proposed methodology through an experimental test-bed based on Commercial-Off-The-Shelf (COTS) technologies, where the experimental results are compared against the theoretical results and assess the pessimism of the theoretical model.

4. An analysis of the impact of the sink mobility on the worst-case network performance and an outline of alternatives for sink mobility

management, namely how the routes must be updated upon the mobility of the sink, and how this procedure affects the worst-case network performance (network inaccessibility times).

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

2.1 Introduction

This chapter gives an overview to the most significant features of the IEEE 802.15.4 standard and ZigBee specification. It particularly focuses on the beacon-enable mode and cluster-tree topology that have ability to provide predictable QoS guarantees for the time-sensitive and energy efficient wireless sensor applications.

IEEE 802.15.4 [9] standard and ZigBee [10] specification stand as the leading communication technologies for large scale, low data rate, low cost and low power consumption Wireless Sensor Networks (WSNs) (In 2012, 802.15.4-enabled chips will reach 292 million, up from 7 million in 2007 [12]). IEEE 802.15.4/ZigBee are quite flexible for a wide range of applications by adequately tuning their parameters (see Chapter 3). They can also provide real-time guarantees for time-sensitive WSN applications (see Chapters 5 and 4). Sometimes, people confuse IEEE 802.15.4 with ZigBee. The IEEE 802.15.4 standard specifies the physical layer and medium access control (MAC) sub-layer, while the network layer and the framework for the application layer are provided by the ZigBee specification such that a full protocol stack is defined. Recently the ZigBee Alliance and the IEEE decided to join forces and ZigBee is the commercial name for the IEEE 802.15.4/ZigBee communication technology.

The IEEE 802.15.4 standard defines two main types of wireless nodes: a *Full-Function Device* (FFD) and a *Reduced-Function Device* (RDF). The FFD implements all the functionalities of the 802.15.4 protocol and can operate in three modes serving as a PAN (Personal Area Network) coordinator, a coordinator, or an end device. On the other hand, the RFD can operate only as an end device using a reduced implementation of the 802.15.4 protocol, which requires minimal resources and memory capacity. An end device must be associated with a coordinator and communicates only with it. Coordinators can communicate with each other, and they are capable to relay messages. One of the coordinators is designed as a PAN coordinator and it holds special functions such as identification, formation and control of the entire network.

The data payload is passed from the application layer to the network layer, and it is referred to as the Network Service Data Unit (NSDU) (Figure 2.1). This payload is prefixed with a network header (NHR) of 64-bit size, which comprises frame control, addressing and sequencing information [10]. The NHR and NSDU form the Network Protocol Data Unit (NPDU), which is passed to the data link layer as the MAC payload, i.e. MAC Service Data Unit (MSDU). The maximum MAC payload that can be transmitted in a data frame is equal to *aMaxMACPayloadSize* (944 bits). The MAC payload is prefixed with a MAC Header (MHR) and appended with a MAC Footer (MFR). The MHR contains the frame control field, data sequence number, addressing fields, and optionally the auxiliary security header. The MFR is composed of a 16-bit frame control sequence (FCS). The MHR, MSDU, and MFR together form the MAC Protocol Data Unit (MPDU).The maximum size of a MPDU is equal to *aMaxPHYPacketSize* (1016 bits). Hence, the minimum size of MAC Header is equal to 56 bits using 16-bit short addresses. The MPDU is passed to the physical layer as the Physical Service Data Unit (PSDU), which becomes the PHY payload. The PHY payload is prefixed with a Physical Header (PHR) of 8-bit size and a Synchronization Header (SHR) of 40-bit size enabling the receiver to achieve symbol synchronization. The SHR, PHR, and PSDU together form the Physical Protocol Data Unit (PPDU), which can be dispatched to a wireless channel.

2.2 Physical layer

The physical layer is responsible for data transmission and reception using a certain radio channel according to a specific modulation and spreading techniques. The IEEE 802.15.4-2003 [13] standard supports three unlicensed frequency bands: 2.4 GHz (worldwide, 16 channels), 915 MHz (North America and some Asian countries, 10 channels) and 866 MHz (Europe, 1 channel). The data rate is 250 kbps at 2.4 GHz, 40 kbps at 915 MHz and 20 kbps at 868 MHz. In addition, four frequency band patterns have been

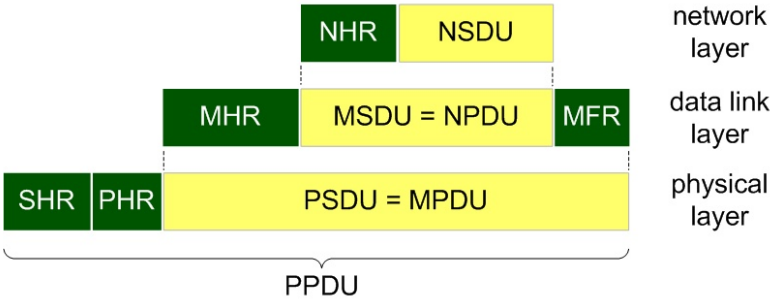


Figure 2.1: Structure of the IEEE 802.15.4/ZigBee frames.

added to the 868/915 MHz bands in the last revision of the standard (IEEE 802.15.4-2006 [9]). All of these frequency bands are based on the Direct Sequence Spread Spectrum (DSSS) or Parallel Sequence Spread Spectrum (PSSS) spreading techniques that have inherently good noise immunity. The standard also allows energy detection, link quality indication, clear channel assessment and radio channel switching.

This thesis only considers the 2.4 GHz band with 250 kbps data rate, which is also supported by the TelosB motes [14] used in the experimental test-beds. In addition, the Zigbee specification is only defined for this frequency band.

2.3 Data link layer

The MAC sub-layer supports the beacon-enabled or non beacon-enabled modes that may be selected by a central controller of the WSN, i.e. PAN coordinator. In non beacon-enabled mode, the nodes can simply transmit messages using unslotted Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) channel access protocol. In fact, the "collision avoidance" mechanism is based on a random delay prior to transmission, which only reduces the probability of collisions. Thus, this mode cannot ensure collision-free and predictable access to the shared wireless medium and, consequently, it cannot provide any time and resource guarantees. On the other side, the beacon-enabled mode enables the synchronization of a WSN using periodic beacon frames, the energy conservation using low duty-cycles, and the provision of collision-free and predictable access to the wireless medium through the *Guaranteed Time Slot* (GTS) mechanism. Thus, when the timeliness and energy efficiency are the main concerns, the beacon-enabled mode should be employed.

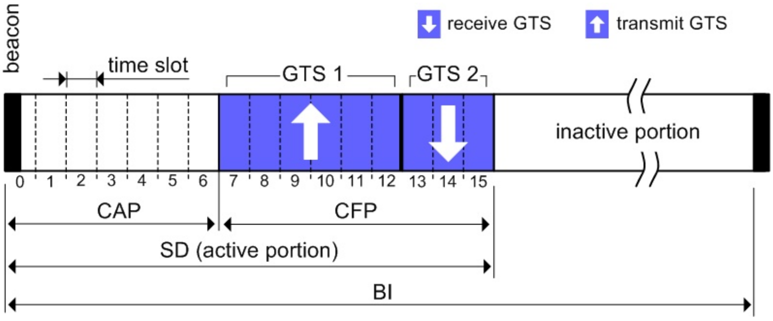


Figure 2.2: Superframe structure of IEEE 802.15.4.

In beacon-enabled mode, beacon frames are periodically sent by a coordinator to synchronize nodes (i.e. coordinators or/and end devices) that are associated to it and to describe the structure of the superframe (Figure 2.2). *Beacon Interval* (BI) is defined as the time interval between two consecutive beacons, and it is divided into an active portion and, optionally, a following inactive portion. During the inactive portion, each associated node may enter a low power mode to save energy resources.

The active portion, corresponding to *Superframe Duration* (SD), is divided into 16 equally-sized time slots, during which data transmission is allowed. These time slots are further grouped into a *Contention Access Period* (CAP) using slotted CSMA/CA for the best-effort data delivery, and an optional *Contention Free Period* (CFP) supporting the time-bounded data delivery. Within the CFP, the coordinator can allocate Guaranteed Time Slots (GTS) to its associated nodes. The CFP supports up to 7 GTSs and each GTS may contain one or more time slots. Each node may request up to one GTS in *transmit direction*, i.e. from the associated node to the coordinator, or/and one GTS in *receive direction*, i.e. from the coordinator to the associated node. A GTS is activated upon a request, where a node explicitly expresses the number of time slots that it wants to allocate from its coordinator. The allocation of the GTS cannot reduce the length of the CAP to less than the value specified by *aMinCAPLength* constant (7.04 ms [9]) ensuring that commands can still be transferred when GTSs are being used. A node to which a GTS has been allocated can also transmit best-effort data during the CAP. Note that there are neither beacons nor superframes in non beacon-enabled mode.

The *explicit GTS allocation*, which is natively supported by the IEEE 802.15.4 standard, has the advantage of being simple. However, the GTS resources may quickly disappear, since a maximum of 7 GTSs can be allocated in each superframe. Moreover, the explicit GTS allocation may be not efficient enough in terms of bandwidth utilization, since the bandwidth of a GTS is given by an integer multiple of the time slot. To overcome these limitations, the implicit GTS Allocation MEchanism (i-GAME) was proposed by Koubâa et al. [15]. The i-GAME approach enables the use of a GTS by several nodes while all their requirements (e.g. bandwidth, delay) are still satisfied. For that purpose, the authors have proposed an admission control algorithm that enables to decide whether to accept a new GTS allocation request or not, based not only on the remaining time slots, but also on the traffic specifications of the flows, their delay requirements and the available bandwidth resources. Hence, more than 7 nodes may be associated to a coordinator. On the other hand, the implicit GTS allocation may enlarge the end-to-end delay.

Beacon Interval (BI) and duration of active portion (SD) are defined by two parameters, the *Beacon Order* (BO) and the *Superframe Order* (SO) as follows:

$$\left.\begin{aligned}\text{BI} &= aBaseSuperframeDuration \cdot 2^{BO} \\ \text{SD} &= aBaseSuperframeDuration \cdot 2^{SO}\end{aligned}\right\} \text{for } 0 \le SO \le BO \le 14 \qquad (2.1)$$

where *aBaseSuperframeDuration* = 15.36 ms (assuming the 2.4 GHz frequency band with 250 kbps data rate) and denotes the minimum duration of the active portion when $SO = 0$. The ratio of the active portion (SD) to the BI is called the *duty-cycle*. IEEE 802.15.4 standard is optimized for low duty-cycles (under 1%), which is interesting for the battery-powered nodes to significantly reduce their power consumption. On the other side, low duty-cycle enlarges the end-to-end delays in multi-hop networks such that a fair trade-off between timeliness and energy efficiency must be found.

The MAC sub-layer needs a finite amount of time to process data received by the physical layer. Hence, the consecutive frames are separated by *Inter-Frame Spacing* (IFS) (Figure 2.3). The IFS is equal to a Short Inter-Frame Spacing (SIFS) of a duration of at least 0.192 ms, for MAC frame (MPDU) lengths smaller than or equal to *aMaxSIFSFrameSize* (144 bits). Otherwise, the IFS is equal to a Long Inter-Frame Spacing (LIFS) of a duration of at least 0.64 ms, for MAC frame lengths greater than *aMaxSIFSFrameSize* bits and smaller than the maximum size of a MAC frame *aMaxPHYPacketSize* (1016 bits). In an acknowledged transmission the IFS follows the acknowledgement frame, otherwise the IFS follows the frame itself (Figure 2.3).

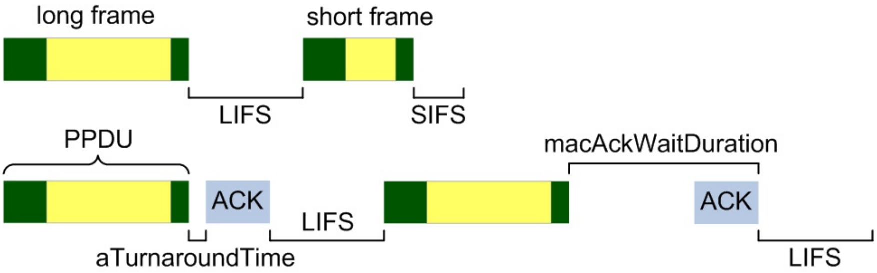


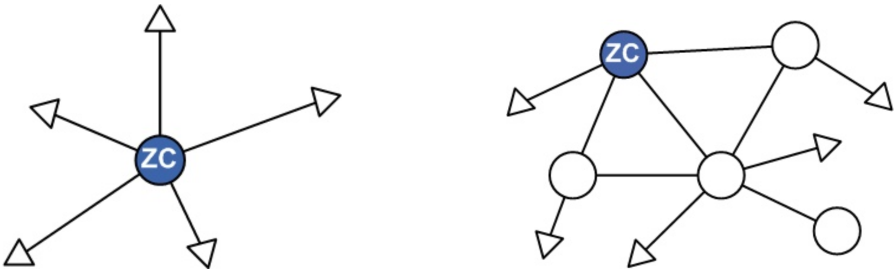
Figure 2.3: The Inter-Frame Spacing.

IEEE 802.15.4 standard also supports acknowledgement and retransmission mechanisms to minimize the communication errors coming from the unreliable and time-varying characteristics of wireless channels. In the case of acknowledged transmissions, the sender waits for the corresponding acknowledgement frame at most *macAckWaitDuration* (0.864 ms). The receiver can commence transmission of acknowledgement frame *aTurnaroundTime* (0.192 ms) after the reception of the data or command frame. If an acknowledgement frame is received within *macAckWaitDuration*, the transmission is considered successful. Otherwise, the data transmission and waiting for the acknowledgement are repeated up to a maximum of *macMaxFrameRetries* (range 0-7, default 3) times. If an acknowledgement frame is not received after *macMaxFrameRetries* retransmissions, the transmission is considered failed. Note that each retransmission decreases the effective throughput and increases the communication delay and energy consumption such that a fair trade-off between reliability of data transmission, timeliness and energy efficiency must be found (see the simulation results in Section 4.6).

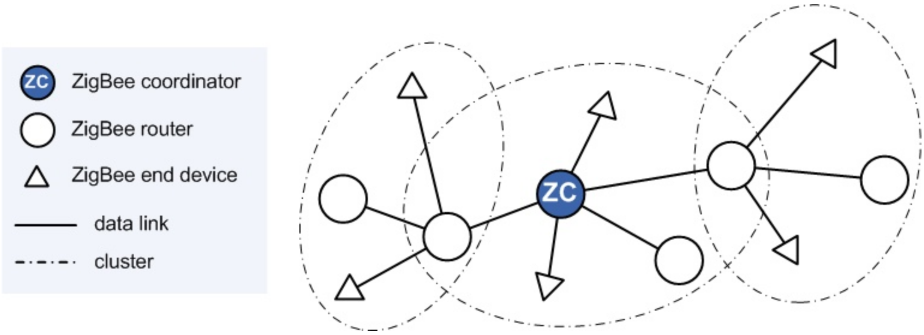
The whole transmission, including the frame, IFS and eventual acknowledgement and retransmissions, must be completed before the end of CAP or the end of the current GTS. Otherwise, it must wait until the next CAP or GTS in the next superframe.

2.4 ZigBee network layer

The ZigBee [10] network layer allows the network to spatially grow using multi-hop communications, without requiring high power transmitters. Responsibilities of the network layer include mechanisms used to associate to and disassociate from a network, apply security to outgoing frames, and routing frames to intended destinations.



(a) star topology (b) mesh topology



(c) cluster-tree topology

Figure 2.4: IEEE 802.15.4/ZigBee network topologies.

Regarding the node's role in the network, ZigBee specification defines three types of nodes: ZigBee coordinator, ZigBee router and ZigBee end device. The node that is capable to directly associate other nodes and can participate in multi-hop routing is referred to as *ZigBee router* (ZR). Any FFD operates in coordinator mode can act as a ZigBee router. An FFD operating in PAN coordinator mode acts as *ZigBee coordinator* (ZC). Every WSN shall include one ZigBee coordinator that holds special functions such as identification, formation and control of the entire network. ZigBee coordinator also participates in routing once the network is formed. The node that does not allow association of other nodes and do not participate in routing are referred to as *ZigBee end device* (ZED). Any FFD or RFD can act as a ZigBee end device.

Star, mesh and cluster-tree are three logical topologies supported in the IEEE 802.15.4/ZigBee as shown in Figure 2.4. While IEEE 802.15.4 standard in the beacon-enabled mode supports only the star topology, the ZigBee specification has proposed its extension to the multi-hop cluster-tree and mesh topologies. In the *star topology*, the communications are centralized and established exclusively between a ZigBee coordinator and its associated ZigBee end devices. 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 associated ZEDs, the ZC emits regular beacon frames. Consequently, each ZED can enter a low power mode to save their energy whenever it is not active. The ZED can also request for the GTS ensuring predictable and contention-free medium access. The main advantages of star topology are its simplicity and predictable and energy efficient behaviour. The drawbacks are limited scalability and ZC as a single point of failure. The ZC's battery resource can be also rapidly ruined since all traffic is routed through ZC. Hence, the star networks are suitable for simple and small scale applications.

Infrastructure-less mesh topology and infrastructure-based cluster-tree topology allow more complex network formations to be implemented. The *mesh topology* differs from the star topology in that the communications are decentralized and any node can directly communicate with any other node within its radio range. The mesh network usually operates in ad hoc fashion that induces unpredictable end-to-end connectivity between nodes. In contrast with the star topology, the mesh topology provides good scalability and enhanced network flexibility such as redundant routing paths that increases end-to-end reliability of data transmission and ensures fair resource usage. In addition, this communication redundancy can eliminate single point of failure. On the other hand, the probabilistic routing protocol (e.g. Ad hoc On Demand Distance Vector (AODV) routing protocol defined in ZigBee) and contention-based MAC protocol cause unpredictable performance and resource bounds. Moreover, since the routing paths cannot be predicted in advance, the nodes cannot enter low power mode which leads to a useless waste of energy.

The *cluster-tree topology* combines the benefits of both above mentioned topologies such as good scalability, network synchronization and predictable and energy efficient behaviour, which is suited for medium-scale time-sensitive applications using battery-powered nodes. Cluster-tree is tree-based topology, where the nodes are organized in logical groups, called clusters. Each router (including ZigBee coordinator) forms a cluster and is referred to as its cluster-head. All nodes associated with a given cluster-head belong to its cluster, and the cluster-head handles all their transmissions. Note that each cluster can be seen as a star subnetwork. ZigBee coordinator is identified as the root of the tree and forms the initial cluster. The other ZigBee routers join the cluster-tree in turn by establishing themselves as cluster-heads, starting to generate the beacon frames for their own clusters. Contrary to the mesh topology, there is a single routing path between any pair of nodes in a cluster-tree topology. Hence, multi-hop communication is deterministic and time efficient because each node only interacts with its predefined set of nearby nodes. The deterministic routing protocol and contention-free medium access (GTS) ensure predictable network performance and resource bounds. In addition, thanks to the deterministic routing and synchronous behaviour in cluster-tree topology, the nodes know their active time in advance. Hence, each node can save its energy by entering the low power mode when it does not participate in the routing. Contrary to the mesh network, the cluster-tree network is less flexible since it relies on the pre-deployed infrastructure. The cluster-tree network also needs specific algorithms to correctly design the parameters that regulate beacon and data transmission in order to achieve a good network capacity. Clearly, the behaviour of the whole cluster-tree network strongly depends on the setting of the parameters. For example, if $SO = BO$ there will be no inactive portion meaning that the nodes cannot enter into the low power mode and, on the other hand, if SO is set too low (and so does the duty-cycle), the data rate has to be decreased.

	star	**mesh**	**cluster-tree**
scalability	no	yes	yes
energy efficiency	yes	no	yes
network synchronization	yes	no	yes
redundant paths	no	yes	no
node mobility	partial	yes	partial
deterministic routing	yes	no	yes
contention-free medium access	yes	no	yes

Table 2.1: Star vs. mesh vs. cluster-tree topologies.

Table 2.1 summarizes the important features of the above mentioned topologies as they are defined in the IEEE 802.15.4/ZigBee. Remind that the star and cluster-tree networks can operate on beacon-enabled mode, which can provide predictable resource guarantees (e.g. bandwidth and buffer size), network synchronization and energy conservation. Note that the beacon-enabled mode is not permitted in mesh networks. In contrast with the star and mesh networks, the cluster-tree network requires precise cluster scheduling (Chapter 4) to avoid inter-cluster collisions (messages/beacons transmitted from nodes in different overlapping clusters). IEEE 802.15.4 standard and Zigbee specification admit the formation of the cluster-tree network but none of them imposes any algorithm or methodology to create or organize it. Thus, the cluster-tree topology expresses several challenging and open research issues in this area, which have been addresses in this thesis.

Chapter 3

IEEE 802.15.4/ZigBee simulation model: delay/throughput evaluation of the GTS mechanism

3.1 Introduction

Simulation and modelling are important approaches to developing and evaluating the systems in terms of time and cost. A simulation shows the expected behaviour of a system based on its simulation model under different conditions. To study system behaviour and performance by means of real deployment or setting up a test-bed may require much effort, time and financial costs. However, the simulation results are not necessarily accurate or representative. Hence, the goal for any simulation model is to accurately model and predict the behaviour of a real system.

Recently, several analytical and simulation models of the IEEE 802.15.4 [9] protocol have been proposed. Nevertheless, currently available simulation models [16] for this protocol are both inaccurate and incomplete, and in particular they do not support the Guaranteed Time Slot (GTS) mechanism, which is required for time-sensitive wireless sensor applications.

This chapter presents an accurate IEEE 802.15.4/ZigBee simulation model developed in the Opnet Modeler simulator [17]. Opnet Modeler was chosen due to its accuracy and to its sophisticated graphical user interface. The idea behind this simulation model was triggered by the need to build a very reliable model of the IEEE 802.15.4 and ZigBee protocols for Wireless Sensor Networks (WSNs). The simulation model is validated with focus on the GTS mechanism using the the Network Calculus based analytical model [11]. The results previously obtained through Network Calculus upper bound or overpass the results obtained through simulation. The tighter simulation results allow to propose a novel methodology to tune the protocol parameters such that a better performance of the protocol can be guaranteed.

Contribution

The motivation that has driven this work was the performance evaluation of the real-time behaviour of the IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs. Thus, this chapter contributes an accurate Opnet simulation model for the IEEE 802.15.4 and ZigBee protocols focusing on the implementation of the GTS mechanism and ZigBee hierarchical routing strategy. The simulation model is used to carry out a set of experiments and to compare the performance evaluation of the GTS mechanism as given by the two alternative approaches, namely simulation and analytical.

Additionally, and probably more importantly, based on the simulation model a novel methodology is proposed to tune the IEEE 802.15.4 parameters (e.g. SO, BO) such that a better performance of the IEEE 802.15.4 protocol can be guaranteed, both concerning maximizing the throughput of the allocated GTS as well as concerning minimizing media access delay.

In particular, the Chapter 3 presents the following contributions:

1. An accurate simulation model of IEEE 802.15.4/ZigBee protocols that has been implemented in the Opnet network simulator.
2. A demonstration of the validity of proposed simulation model through an analytical model based on Network Calculus.
3. A novel methodology to tune the IEEE 802.15.4 parameters such that a better performance can be guaranteed.

3.2 Related work

Opnet Modeler, ns-2 and OMNeT++ are widely used and popular network simulators which, among others, include a simulation model of the IEEE 802.15.4 protocol. Of course, each simulator has its own disadvantages and advantages. The 802.15.4/ZigBee simulation model in Opnet model library [17] supports only non beacon-enabled mode, therefore, the cluster-tree topology and GTS mechanism cannot be simulated. In addition, the source codes of the network and application layers are not available. The National Institute of Standards and Technology (NIST) has developed own Opnet simulation model for the IEEE 802.15.4 protocol [18]. However, while that model implements the slotted and the unslotted CSMA/CA MAC protocols it does not support the GTS mechanism as well. It also uses its own radio channel model rather than the accurate Opnet wireless library. The Network Simulator 2 (ns-2) [19] is an object-oriented discrete event simulator including a simulation model of the IEEE 802.15.4 protocol. The accuracy of its simulation results is questionable since the MAC protocols, packet formats, and energy models are very different from those used in real WSNs [20]. This basically results from the facts that ns-2 was originally developed for IP-based networks and further extended for wireless networks. Moreover, the GTS mechanism was not implemented in the ns-2 model. OMNeT++ (Objective Modular Network Test-bed in C++) [21] is another discrete event network simulator supporting unslotted IEEE 802.15.4 CSMA/CA MAC protocol only. Finally, note that while ns-2 and OMNeT++ are open source projects, the Opnet Modeler is commercial project providing a free of charge university program for academic research projects.

There have also been several research works on the performance evaluation of the IEEE 802.15.4 protocol using simulation model. Zheng et al. [22] have evaluated various features of the 802.15.4 protocol (e.g. direct, indirect and GTS data transmissions), and investigated the collision behaviour of IEEE 802.15.4. In addition, the simulation experiments compare the performance of 802.15.4 and 802.11 (WiFi) protocols. The authors have developed own ns-2 simulation model of 802.15.4 protocol, which additionally implements beacon-enabled mode and GTS mechanism. Since the network layer has not been implemented, a star topology is only supported. Based on this implementation, Chen et al. [23] have developed own simulation model of IEEE 802.15.4 protocol in OMNeT++. Contrary to the standard OMNeT++ model, their simulation model implements a battery module, beacon-enabled mode and GTS mechanism, and supports only star topology. Using this simulation model, the IEEE 802.15.4 star network has been evaluated in terms of energy consumption and end-to-end communication performance in [24]. Hurtado-Lopez et al. [25] have extended the above mentioned IEEE 802.15.4 model in OMNeT++ to support cluster-tree topology.

In [26], the authors have presented a simulation study of the slotted CSMA/CA MAC protocol deployed by the IEEE 802.15.4 protocol in beacon-enabled mode, using the previous version of the Opnet simulation model. In this chapter, this version has been extend to include the GTS mechanism and ZigBee network layer, which allows a simulation study of the real-time behaviour of cluster-tree WSNs.

3.3 Simulation model

This section presents the structure of the IEEE 802.15.4/ZigBee simulation model [27] that was implemented in the Opnet Modeler simulator.

3.3.1 Simulation model structure

The Opnet Modeler [17] is a commercial discrete-event network modelling and simulation environment, which provides tools for all phases of a system analysis and testing cycle including model design, simulation, data collection and data analysis. Both behaviour and performance of modelled systems can be analysed and visualized in a rich integrated graphical environment. Opnet's Standard Model Library supports hundreds of generic or vendor-specific protocols and technologies that can be use to build the networks. In addition, Opnet Modeler simulator includes a hierarchical development environment to enable modelling of any type of custom protocol and device. The development environment consists of three hierarchical modelling domains (Figure 3.1). Network domain describes network topology in terms of nodes and links. Internal architecture of a node is described in the node domain. Within the process domain, the behaviour of a node is defined using state transition diagrams. Operations performed in each state or transition are described in embedded C/C++ code blocks. The IEEE 802.15.4/ZigBee simulation model builds on the wireless module, an add-on that extends the functionality of the Opnet Modeler with accurate modelling, simulation and analysis of wireless networks.

The IEEE 802.15.4/ZigBee Opnet simulation model implements physical layer and medium access control sub-layer defined in IEEE 802.15.4 [9] standard, and network layer defined in ZigBee [10] specification. The latest version of simulation model [27] supports the following features:

- beacon-enabled mode (beacon frame generation)
- star and cluster-tree topologies
- computation of the power consumption (MICAz and TelosB motes are supported)
- physical layer characteristics
- slotted CSMA/CA MAC protocol

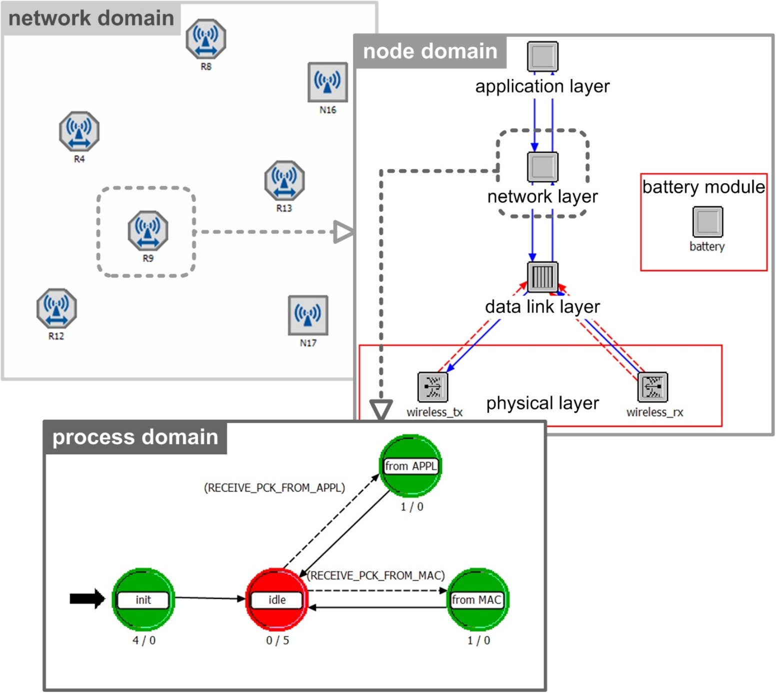


Figure 3.1: The structure of the IEEE 802.15.4/ZigBee Opnet simulation model.

- Guaranteed Time Slot (GTS) mechanism (GTS allocation, deallocation and reallocation functions)
- generation of the acknowledged or unacknowledged best-effort application data (MSDU) transmitted during the CAP
- generation of the acknowledged or unacknowledged real-time application data transmitted during the CFP
- ZigBee hierarchical tree routing
- verification of node's address that must correspond to the ZigBee hierarchical addressing scheme

In accordance to the ZigBee [10] specification, there are implemented three types of nodes in the simulation model, namely a ZigBee coordinator, a ZigBee router and a ZigBee end device. All types of nodes have the same internal architecture (node domain) but they differ in the available user-defined attributes (Section 3.3.2).

The structure of the IEEE 802.15.4/ZigBee simulation model is presented in Figure 3.1. The the *physical layer* consists of a wireless radio transmitter and receiver compliant to the IEEE 802.15.4 [9] standard running at 2.4 GHz frequency band with 250 kbps data rate. Default settings are used for the physical characteristics of the radio channel such as background noise and interference, propagation delay, antenna gain, and bit error rate.

The *data link layer* supports the beacon-enabled mode (non beacon-enabled mode is not supported yet) and implements two medium access control protocols according to the IEEE 802.15.4 standard, namely the contention-based slotted CSMA/CA and contention-free GTS. MAC payload (MSDU) incoming from the network layer is wrapped in MAC header and MAC footer and stored into two separate FIFO buffers, namely a buffer for best-effort data frames and another buffer for real-time data frames. The frames are dispatched to the network when the corresponding CAP or CFP is active. On the other hand, the frame (MPDU) incoming from the physical layer is unwrapped and passed to the network layer for further processing. The data link layer also generates required commands (e.g. GTS allocation, deallocation and reallocation commands) and beacon frames when a node acts as PAN coordinator or router.

The *network layer* implements address-based tree routing (a mesh routing is not supported yet) according to the ZigBee [10] specification. The frames are routed upward or downward along the cluster-tree topology according to the destination address by exploiting the hierarchical addressing scheme provided by ZigBee [10]. This addressing scheme assigns an unique address to each node using the symmetric hierarchical addressing tree given by three parameters, namely the maximum number of children (i.e. routers and end devices) that a router or a coordinator may have (C_m), the maximum depth in the topology (L_m), and the maximum number of routers that a router or a coordinator may have as children (R_m).

The *application layer* can generate unacknowledged and/or acknowledged best-effort and/or real-time data frames transmitted during CAP or CFP, respectively. There is also a *battery module* that computes the consumed and remaining energy levels. The default values of current draws are set to those of the widely-used MICAz [28] or TelosB [14] motes.

3.3.2 User-defined attributes

This section depicts some important user-defined attributes relating to the GTS mechanism and the real-time data traffic. All attributes are described in the reference guide [29] in details.

A coordinator and each router may accept or reject the GTS allocation request from its children according to the value of the attribute *GTS Permit*. Each node (except the coordinator) can specify the time when the GTS

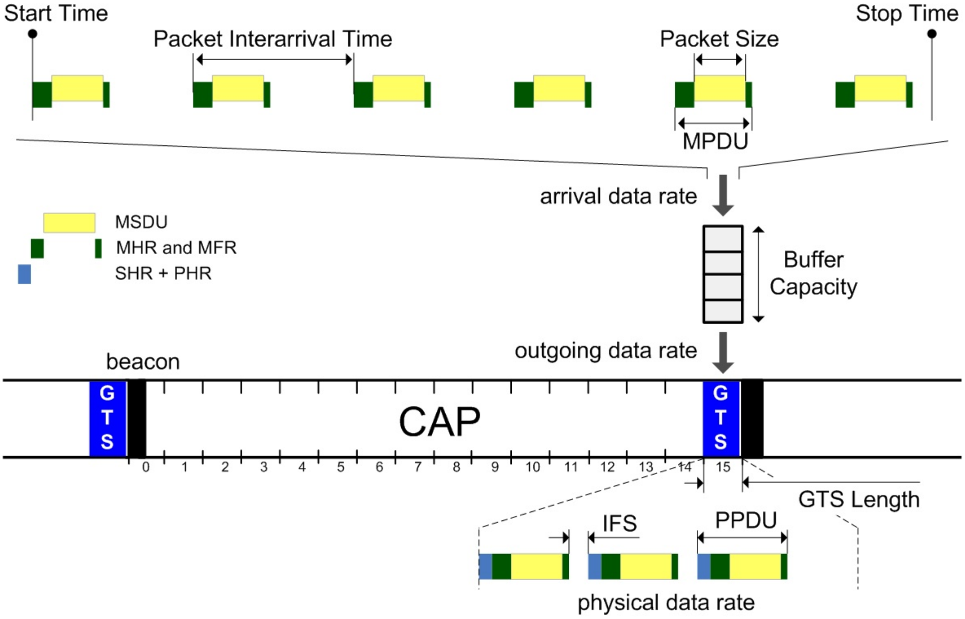


Figure 3.2: The user-defined attributes of the GTS mechanism.

allocation request (GTS *Start Time* attribute) and deallocation request (GTS *Stop Time* attribute) are dispatched to its parent. The allocation request includes the number of required time slots (*Length* attribute) and the transmit or receive direction.

Each node can generate data frames inside the time interval given by *Start Time* and *Stop Time* attributes. The size of frame payload (MSDU) is defined by the *Packet Size* attribute. The *Packet Interarrival Time* attribute defines the inter-arrival time between two consecutive frames. The frames are stored in a buffer. The frames exceeding the buffer capacity (*Buffer Capacity* attribute) are dropped. When the requested GTS is active, a frame (MPDU) is removed from the buffer, prefixed with the headers (SHR and PHR), and it is dispatched to the network with an outgoing data rate equal to physical data rate (250 kbps).

The behaviour and user-defined attributes of the GTS mechanism are shown in Figure 3.2. Other internal protocol parameters use default values specified in the IEEE 802.15.4 standard.

3.4 Simulation setup

In this chapter, a simple star network containing a coordinator and one associated end device is considered. This configuration is sufficient for the performance evaluation of the GTS mechanism, since there is no medium access contention inside the GTS. Thus, having additional nodes would have no influence on the simulation results. Next, the unacknowledged transmission ($macMaxFrameRetries = 0$) is only considered for comparative purposes with the analytical results obtained in [11].

For the sake of simplicity, and without loss of generality, we assume the allocation of only one time slot GTS in transmit direction and a 100% duty-cycle (i.e. $SO = BO$). In what follows, the change of the SO means that the BO also changes while satisfying $SO = BO$. This means that the optional inactive portion is not included in the superframe.

Reliability of data transmission may be enhanced by keeping the frame size as small as practical, as this gives the highest probability of a frame being delivered in the presence of interference. Prolonged battery life is achieved by minimizing the on duration of the radio (receive and transmit modes), where most power is consumed [30]. A small frame size and infrequent transmission both help to achieve this. Hence, small frame sizes are used during the simulation (i.e. *Packet Size* = 40 or 41 bits).

The statistical data (e.g. average, maximum, minimum delays) are computed from a set of 1000 samples. Hence, the simulation time of one run is equal to the duration of 1000 superframe periods and, consequently, the simulation time depends on the Superframe Order (SO).

3.4.1 Simulation vs. analytical models

In Section 3.5, the performance of the GTS mechanism from the Opnet simulation model is evaluated against the analytical model of the GTS mechanism proposed in [11], which is based on the Network Calculus formalism. Network Calculus (Section 5.3) is a mathematical methodology based on min-plus algebra that applies to the deterministic analysis of queuing/flows in communication networks.

The Network Calculus based analytical model relies on the affine arrival curve and rate-latency service curve [31]. This means that each generated application data flow has a cumulative arrival function $R(t)$ upper bounded by the affine arrival curve $\alpha_{b,r}(t) = b + r \cdot t$, where b denotes the burst tolerance and r denotes the average arrival rate. The analytical model is bit-oriented, which means that the application data are generated as a continuous bit stream with data rate r. On the other side, the simulation model has a more realistic frame-oriented basis, where the frames with a specified size are generated with a given period (refer to Figure 3.2). Consequently, the burst tolerance b and arrival rate r, as defined in the analytical model, should be implemented in the simulation model in the following way. A FIFO buffer with a specified capacity substitutes a data burst with a given size, and the arrival data rate is defined as follows:

$$r = \frac{Packet\ size}{Packet\ Interarrival\ Time}\ \text{[bps]} \tag{3.1}$$

The smallest data unit in the analytical model is a bit, while in the simulation model it is a frame with a bounded size.

3.5 Performance evaluation

This section shows how the Superframe Order, the arrival data rate, the buffer capacity and the size of the frame payload impact the throughput of the allocated GTS and the media access delay of the transmitted frames.

3.5.1 Impact of the Superframe Order on the GTS throughput

Throughput as a function of the arrival data rate

The purpose of this section is to evaluate and compare the data throughput during one time slot GTS, for different values of the Superframe Order and for different arrival rates. For a given SO, the data throughput is related to the time effectively used for data transmission inside the GTS. Since the frames are transmitted without acknowledgement, the wasted bandwidth can only result from IFS or waiting for a new frame if the buffer is empty, as depicted in Figure 3.3.

The frames can be dispatched at the physical data rate (250 kbps) if the buffer does not become empty before the end of GTS (Figure 3.3a, b). Otherwise, if the buffer becomes empty, the frames are not stored in the buffer but they are directly dispatched to the network according to their arrival data rate Eq. (3.1), which is often lower than the physical data rate (Figure 3.3c).

Figure 3.4 plots the average data throughput of allocated one time slot GTS for different SOs (with a duty-cycle equal to 1) as a function of the

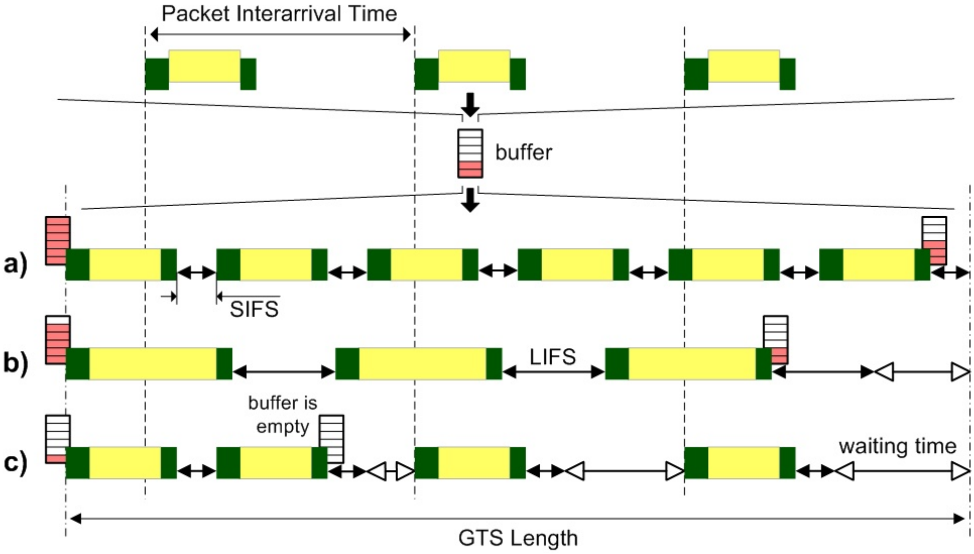


Figure 3.3: The utilization of the GTS bandwidth.

arrival data rate, for two sizes of frame payload (40 and 41 bits). To identify the impact of the arrival data rate on the throughput, the buffer capacity is fixed to 2 kbits.

To show the impact of the IFS on the GTS throughput, the size of the frame payload is set to 40 and 41 bits. When the frame payload size is smaller or equal to 40 bits (MPDU is equal to 144 bits assuming MHR of 88-bit size), the SIFS (48 bits) is used. Otherwise, if the frame payload size is greater or equal to 41 bits, then the LIFS (160 bits) is used. Note that, one additional bit in the frame payload causes 112 additional bits in the IFS. It can be easily observed in Figure 3.4 that the impact of the IFS on the wasted bandwidth is more significant for low SO values.

When the size of the frame payload (*Packet Size*) is fixed, the inter-arrival time (*Packet Interarrival Time*) has to be changed according to Eq. (3.1) in order to reach the required arrival data rates (see Table 3.1). For instance, to achieve 5 kbps arrival data rate, the frame payload with 40 bits size has to be generated every 28.8 ms. We use the same settings as in the analytical model [11] and the *Packet Size* and *Packet Interarrival Time* attributes have been configured as constant values that correspond to the required data rates during each simulation run.

The behaviour of the throughput for low SO values and the lowest arrival data rate (5 kbps) is quite different from the rest of the experiments. This

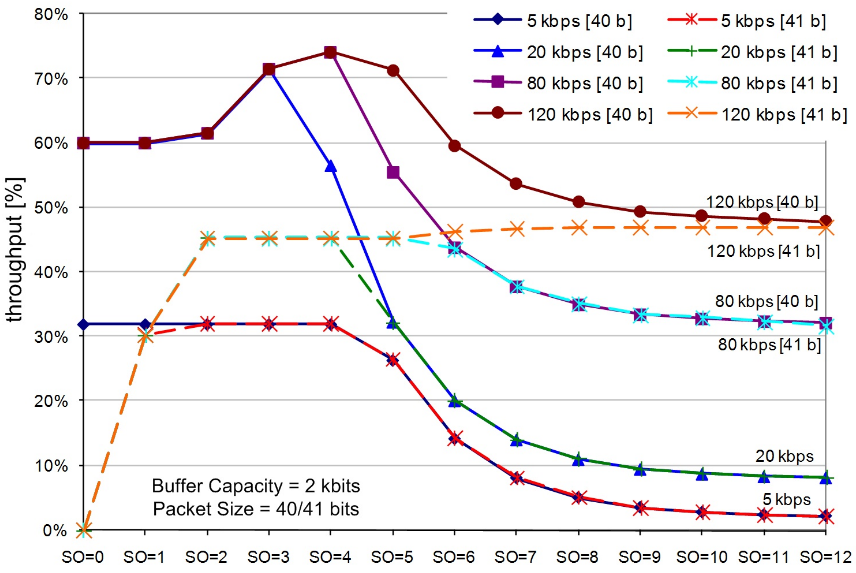


Figure 3.4: GTS throughput as a function of the arrival data rate.

occurs since the Superframe Duration for $SO = 0$ is equal to 15.36 ms, but for 5 kbps arrival data rate the frame payload is generated every 28.8 ms. Thus, in every two superframes, one of them has no available frame in the buffer, and therefore the throughput is roughly the half of the ones resulting from other arrival data rates, where at least one frame is available in the buffer every superframe.

If the size of the frame payload is equal to 41 bits, it results that for $SO = 0$ the throughput is zero for all arrival data rates, since the whole

arrival data rate [kbps]	Packet Interarrival Time [ms]	
	Packet Size = 40 bits	Packet Size = 41 bits
5	28.8	29
10	14.4	14.5
20	7.2	7.25
40	3.6	2.625
80	1.8	1.8125
120	1.2	1.2083

Table 3.1: Relation between arrival data rate r and *Packet Interarrival Time* attribute.

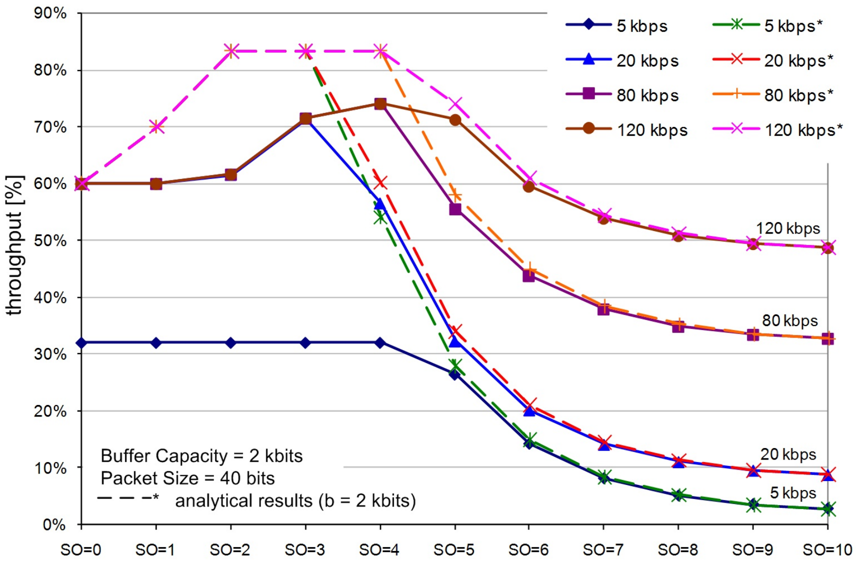


Figure 3.5: GTS throughput as a function of the arrival data rate: simulation vs. analytical model.

transmission (including the frame and LIFS) cannot be completed before the end of the GTS.

For low *SO* values, the throughput grows since the buffer does not become empty during a GTS duration (Figure 3.3a, b), and a significant amount of bandwidth is wasted by the IFS. On the other hand, the throughput for high *SO* values falls, since the buffer becomes empty before the end of the GTS (Figure 3.3c). For a large GTS, a significant amount of bandwidth is wasted when waiting for the incoming frame payload from the application layer. The throughput for high *SO* increases with the arrival data rate (i.e. lower *Packet Interarrival Time*), since the waiting time for the incoming frame payload decreases. It can be easily observed that the throughput performance for high *SO* values is identical and independent of the size of the frame payload.

Analytical results versus simulation results. In Figure 3.5, the analytical and the simulation models have a very similar behaviour in terms of the GTS throughput as a function of the arrival data rate. The throughput performance for high *SO* values has identical values and shape for both models. The simulation results are influenced by the frame-oriented approach of the simulation model, which is more significant for low *SO* values. The analytical model is bit-oriented, therefore it saturates available transmission bandwidth and therefore the throughput performance of the simulation model

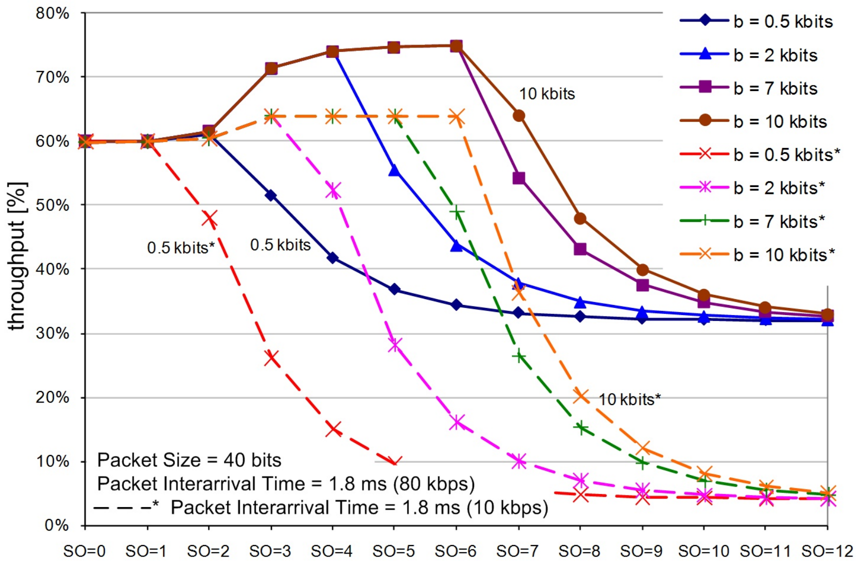


Figure 3.6: GTS throughput as a function of the buffer capacity.

is upper bounded by the maximum throughput of the analytical model (analytical results are drawn with dashed lines).

Throughput as a function of the buffer capacity

Figure 3.6 plots the GTS throughput as a function of the buffer capacity. It can be observed that the throughput increases with the buffer capacity. The highest utilization of the GTS is achieved for SO between 2 to 5.

For the lowest SO values, the throughput depends neither on the arrival data rate nor on the buffer capacity, since the number of incoming frames during a Superframe Duration is low but still sufficient for saturating the GTS. For the highest SO values, the throughput does not depend on the buffer capacity and the throughput values grow with the arrival data rate. This occurs since the buffer becomes empty at the beginning of a large GTS and then, the generated frames are directly forwarded to the network with the rate equal to the arrival data rate.

Analytical results versus simulation results. In Figure 3.7, the behaviours of the analytical and simulation models are very similar in terms of the GTS throughput as a function of the burst size/buffer capacity. The analytical results published in [11] are obtained for the arrival rate equal to

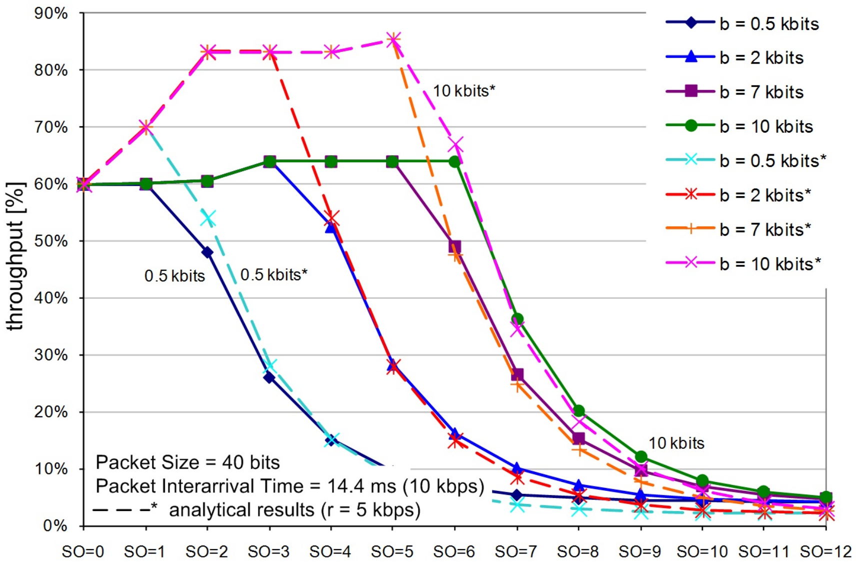


Figure 3.7: GTS throughput as a function of the buffer capacity/burst size: simulation vs. analytical model.

5 kbps. The same arrival rate cannot be used for the simulation, because the lowest data rate 5 kbps has a specific behaviour in case of the simulation model (Figure 3.4). According to the Figure 3.4, an arrival data rate of 10 kbps is selected as the closest one. The throughput performance for high *SO* values has identical values for both models, but for low *SO* values the simulation results are influenced by the frame vs. bit-oriented approach of the simulation and analytical models, as reported for the results given in Figure 3.5. The analytical results upper bound the simulation results (analytical results are drawn with dashed lines).

The first conclusion concerning the GTS throughput for low arrival data rates and low buffer capacities is that high *SO* values are not suitable for ensuring efficient usage of the GTS in terms of data throughput. The maximum utilization of the allocated GTS is achieved with low *SO*s (3–4). The wasted GTS bandwidth increases with *SO*. To avoid this underutilization of the shared wireless medium, the i-GAME mechanism presented in [15] can be used.

3.5.2 Impact of the Superframe Order on the media access delay

In time-sensitive applications, it is necessary to upper bounds the media access delay of the frame. The *media access delay* is defined as the time between the instant when the frame payload is generated at the application layer and the instant when the frame is dispatched to the network. In what follows, this section presents the impact of the SO values on the media access delay of the frame for 100% duty-cycle. The most suitable SO values for providing the lowest delay bound are also determined. Two initial states of the buffer, namely empty or full, are consider.

Figure 3.8 presents the media access delay as a function of the arrival data rate, for a frame payload size and a buffer capacity equal to 40 bits and 4 kbits, respectively. Observe that the media access delay depends neither on the arrival data rate nor on the initial size of the buffer for higher values of arrival data rate (20–120 kbps). In this case the behaviour is almost identical to each other and the lowest delay bound is achieved for SO values equal to 2–3. This occurs since for low SO values ($SO < 5$) the maximum delay is achieved for full buffer. For increased values of the arrival data rate, only the time for filling the buffer grows. This explains also the identical behaviour for initially full or empty buffer. For SO values higher or equal to 5, all frames stored in the buffer (with capacity equal to 4 kbits) can be transmitted during a GTS and the media access delay grows with SO. The value of this breakpoint depends on the buffer capacity (Figure 3.9).

The delay behaviour for the lowest arrival data rate is a monotonic function with the minimum for $SO = 0$. The arrival data rate is too slow and the buffer becomes always empty during a GTS for all SO values. Thus, the value of media access delay grows with SO and does not depend on the buffer capacity. When the buffer is initially full, the maximum delay is achieved at the beginning, and then the buffer becomes gradually empty. For $SO \geq 6$, the buffer is filled up during a Superframe Duration and the behaviours of initially full and empty buffers are met. The specific delay behaviour, for an arrival data rate equal to 10 kbps, is explained in more details, with the support of the results shown in Figure 3.9.

Figure 3.9 shows the media access delay of the frame as a function of the buffer capacity, for a frame payload size and an inter-arrival time equal to 40 bits and 14.4 ms (i.e. $r = 10$ kbps), respectively. For the low SO values (0 and 1), the number of generated frames during a Superframe Duration is higher than the maximal number of potentially transmitted frames during the GTS so that the number of stored frames in the buffer grows. The maximum delay

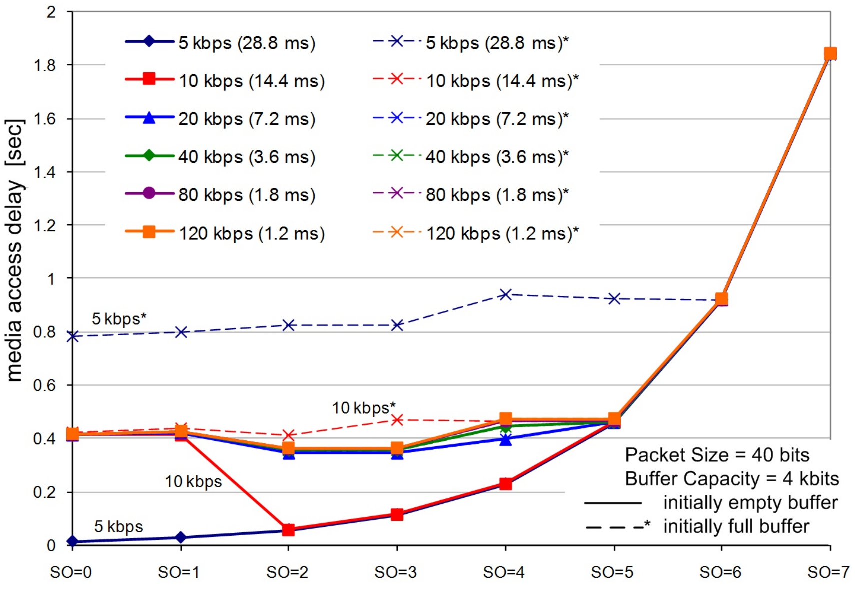


Figure 3.8: The media access delay as a function of the arrival data rate.

of the frame is reached when the buffer is full. Therefore, the media access delay depends only on the buffer capacity and grows with it. For increasing SO values, only the time when the buffer will be full grows. The delays are roughly constant since when the SD is doubled, (i.e. SO value is incremented by one) the GTS duration has to be doubled too. In what follows, the number of generated and transmitted frames is also doubled, thus their ratio stays constant.

When the buffer is initially empty and SO values are higher than 2, the media access delay depends only on the SO values instead of the buffer capacity, and it is roughly equal to the SD minus one time slot GTS duration. This occurs since the number of generated frames is lower than the maximum number of potentially transmitted frames so that no frame is stored in the buffer between two consecutive GTSs. When the buffer is initially full, the media access delay still depends on the buffer capacity until the value of SO causes that the full buffer becomes empty during a GTS. Afterwards, the delay depends only on the value of SO. The maximum delay is reached at the beginning, before the buffer becomes empty.

In this special case, for the lowest buffer capacity (0.5 kbits), the media access delay function is monotonic and grows with SO values, which makes $SO = 0$ the most suitable for providing the lowest delay. For higher buffer

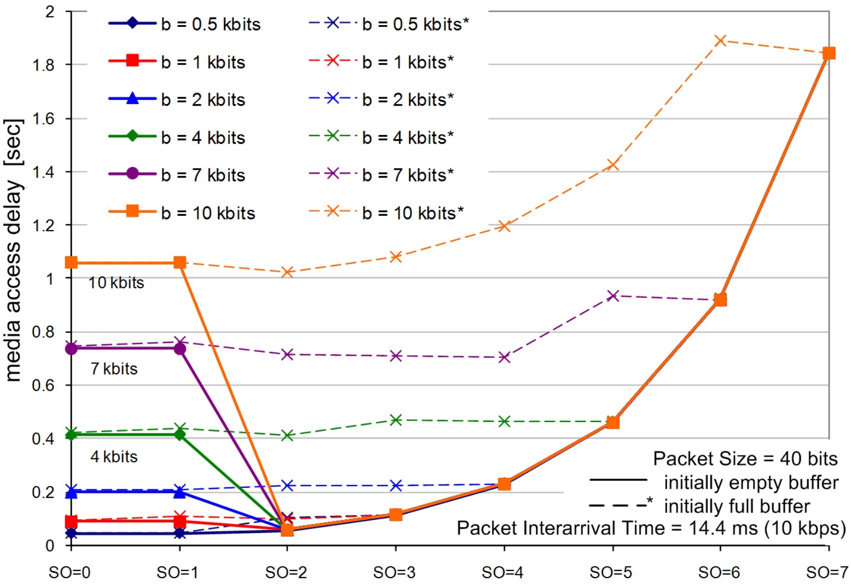


Figure 3.9: The media access delay as a function of the buffer capacity.

capacities, the most suitable value of SO in terms of the lowest delay is definitely 2 and does not depend on the buffer capacity, when the buffer is initially empty.

For the next experiments, the initially empty buffer is only considered. The average and maximum media access delays as a function of the buffer capacity are compared in Figure 3.10.

The maximum delay is achieved at the beginning of each GTS for the first frame removed from the buffer (see Figure 3.11). The following frames removed from the buffer during the GTS have lower delays than the first one, since the arrival data rate is often lower than the outgoing data rate (equal to 250 kbps). For low SO values, the number of dispatched frames during a GTS is also low, and the average delay is then close to the maximum delay. For high SO values, the difference in delay between the first and last frames removed from the buffer during a GTS is greater, and the average delay is then further from the maximum delay.

Analytical results versus simulation results. The simulation and analytical results of the media access delay as a function of the buffer capacity or burst size b are compared in Figure 3.12. The analytical results are obtained for the arrival data rate equal to 5 kbps. The same arrival data rate cannot be used for the simulation model, because the delay for arrival

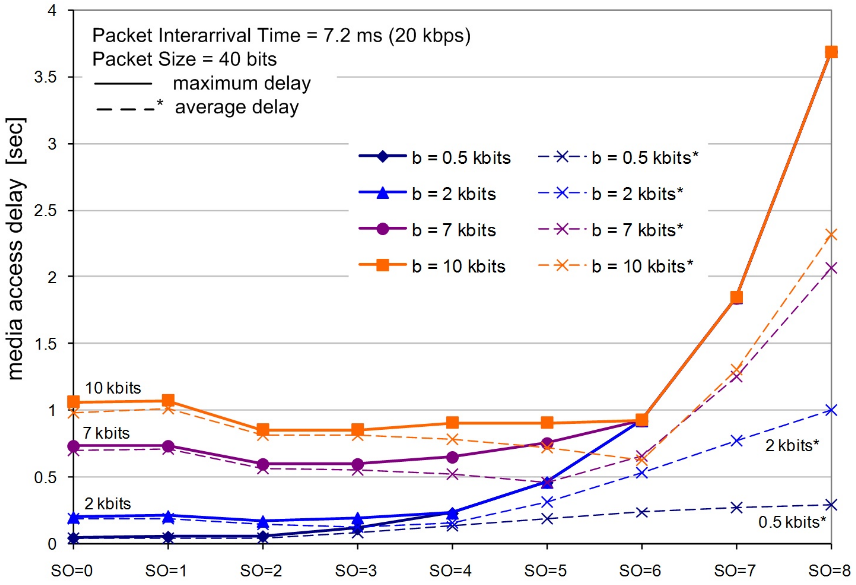


Figure 3.10: Average vs. maximum media access delay as a function of the buffer capacity.

data rate equal to 5 kbps does not depend on the buffer capacity (Figure 3.8). According to the results shown in Figure 3.8, the arrival data rate equal to 20 kbps has been selected as the closest one. Hence, we cannot compare the values, but only the behaviour of the models in terms of media access delay. This behaviour is roughly similar for both models, and the lowest delay is achieved for $SO = 2$ for the case of higher buffer capacity (2–10 kbits), or for $SO = 0$ in case of lower buffer capacity (0.5 and 1 kbits). The difference between frame-oriented and bit-oriented approaches of the simulation and analytical models, respectively, can be observed for the higher SO values. In case of the analytical model, the delay curves converge slowly into a single one (analytical results are drawn with dashed lines).

In summary, WSN applications with low data rates and low buffer capacities achieve the lowest delay bound for $SO = 0$. However, for higher buffer capacities (more than 1 kbits) and higher arrival data rates (more than 10 kbps) the most suitable value of SO for providing real-time guarantees is 2. The simulation and analytical results are roughly identical in terms of the media access delay, and the simulation results are upper bounded by the analytical results.

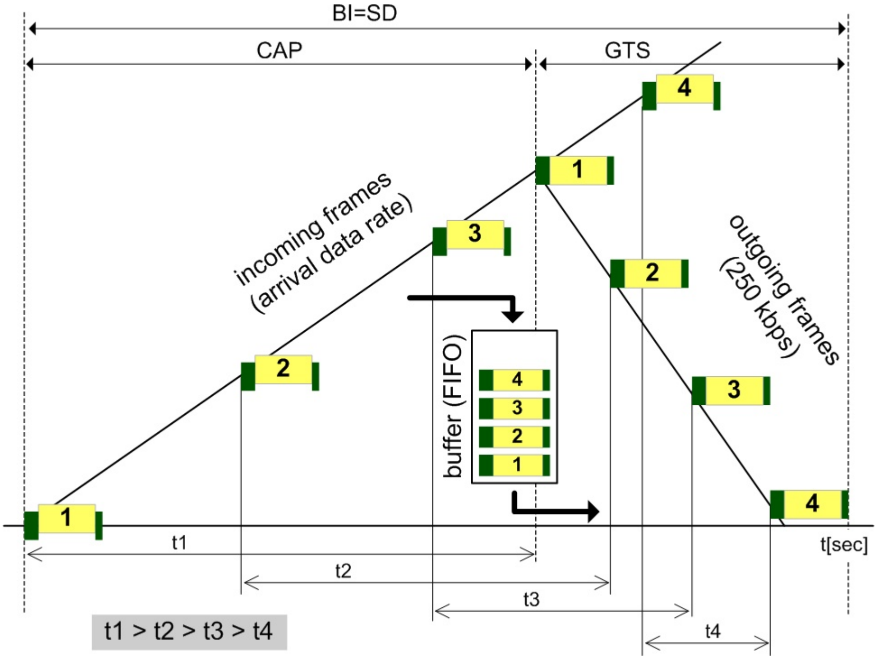


Figure 3.11: Delay of the frames stored in the FIFO buffer.

3.6 Conclusions

This chapter briefly describes an Opnet simulation model of the IEEE 802.15.4/ZigBee protocols focusing on the GTS mechanism and ZigBee hierarchical routing strategy. The IEEE 802.15.4/ZigBee Opnet simulation model is made available publicly in open source [27].

This chapter particularly focuses on the performance evaluation of the GTS mechanism, comparing the obtained simulation results with the ones that were previously obtained [11] using an analytical model based on Network Calculus. The behaviours of both models are roughly identical in terms of the GTS data throughput and the media access delay, and the analytical results are more pessimistic than the simulation results. Discrepancies (most significant for low SOs) are mainly due to the impact of the bit-oriented and frame-oriented approaches used by the analytical and simulation models, respectively.

An optimal setting of the IEEE 802.15.4 GTS mechanism for obtaining maximum data throughput and minimum access delay has also been proposed. For applications with low data arrival rates and low buffer capacities, the maximum utilization of the allocated GTS is achieved for

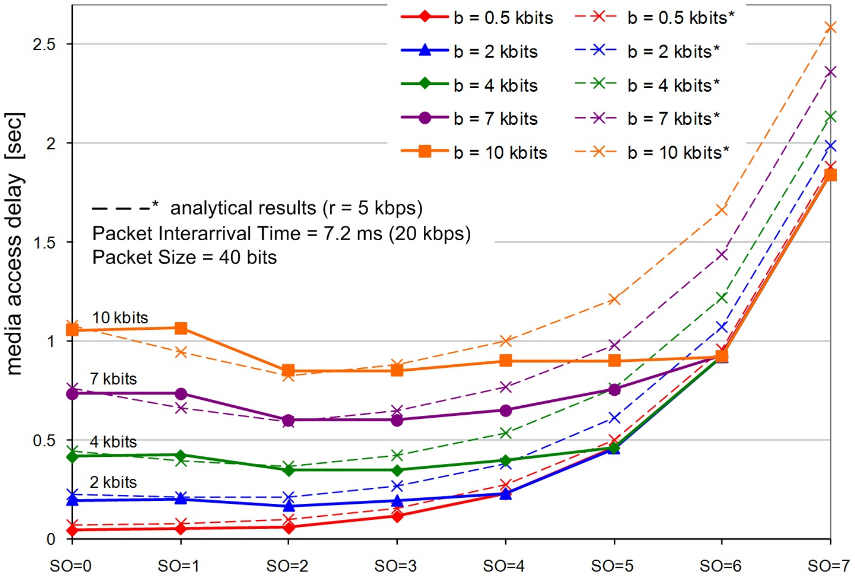


Figure 3.12: The media access delay as a function of the buffer capacity/burst size: simulation vs. analytical model.

low *SO*s (3–4). However, the *SO* equal to 2 is the most suitable value for providing real-time guarantees in time-sensitive WSNs, since it grants the minimum access delay for the GTS frames. High *SO*s are not suitable for ensuring efficient usage of the GTS neither in terms of data throughput nor media access delay.

The future work includes the implementation of the non beacon-enabled mode and support for mesh routing protocols.

Chapter 4

Energy efficient scheduling for cluster-tree Wireless Sensor Networks with time-bounded data flows

4.1 Introduction

This chapter assumes a static deployment of wireless nodes organized in the cluster-tree topology along with the set of periodic time-bounded flows (each given by parameters such as sink node, source nodes and end-to-end deadlines), which must be known in network design time. All nodes may have sensing or/and actuating capabilities, therefore they can be sources or/and sinks of data flows. Note that the cluster-tree network is considered to already being set up, i.e. each node knows its parent and child nodes (e.g using the ZigBee tree addressing scheme [10]).

In cluster-tree WSNs, the flows traverse different clusters on their routing paths from the source nodes to the sink nodes. The clusters may have collisions when they are in the neighbourhood. Thus, the key problem solved in this chapter is to find a periodic schedule, called *Time Division Cluster Schedule* (TDCS), which specifies when the clusters are active while avoiding possible inter-cluster collisions and meeting all data flows' end-to-end deadlines. The fact that the cluster is active only once during the schedule period [10] leads to so called cyclic behaviour of periodic schedule (i.e. time between the instant when a source sends the message and the instant when the sink receives this message spans over several periods) when there are the flows with opposite direction in a WSN. Hence, the TDCS is characterized not only by the moments when the clusters become active within the period, but due to the cyclic nature of the problem it is also characterized by the index of the period for each flow in a given cluster. Since wireless nodes usually use battery for energy supply, the objective is also to minimize the energy consumption of the nodes by maximizing the schedule period (consequently maximizing time when the nodes stay in low power mode).

The interdependence of reliability, energy consumption and timeliness introduces additional complexity to the network design. Thus, this chapter also provides a simulation analysis of how the maximum number of retransmission impacts the reliability of data transmission, the energy consumption of the nodes and the end-to-end communication delay in the IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs. The simulation study is based on the simulation model that was implemented in the Opnet Modeler (Section 3.3). The configuration parameters of the network are obtained directly from the proposed TDCS scheduling tool.

Contribution

The main outcome of this chapter is the provision of a Time Division Cluster Scheduling (TDCS) mechanism based on the cyclic extension of RCPS/TC (Resource Constrained Project Scheduling with Temporal Constraints) problem for a cluster-tree WSN, assuming bounded communication errors. The objective is to meet all end-to-end deadlines of a predefined set of time-bounded data flows while minimizing the energy consumption of the nodes by setting the TDCS period as long as possible. The performance evaluation of the TDCS scheduling tool shows that the problems with hundreds of nodes can be solved while using optimal solvers. The scheduling tool enables system designers to efficiently configure all the required parameters of the IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs in the network design time. The practical application of proposed scheduling tool for the configuration of the IEEE 802.15.4/ZigBee cluster-tree WSNs is demonstrated through the simulation study.

In particular, the Chapter 4 presents the following contributions:

1. A formulation of the scheduling problem by a cyclic extension of RCPS/TC (Sections 4.4.2 and 4.4.3). Using this formulation, the users are not restricted to a particular implementation but they can make a similar extension to any of the algorithms solving this type of problem.

2. A solution of cyclic extension of RCPS/TC by an Integer Linear Programming (ILP) (Section 4.4.4), where a grouping of Guaranteed Time Slots (GTS) leads to very efficient ILP model having a few decision variables.
3. An application of this methodology to a specific case of IEEE 802.15.4/ ZigBee cluster-tree WSNs, as an illustrative example that confirms the applicability of general approach for specific protocols (Section 4.4.1).
4. A time complexity evaluation of the cluster scheduling algorithm implemented in Matlab while using the simplex-based GLPK solver (Section 4.5).
5. A simulation analysis of how the maximum number of retransmissions impacts the reliability of data transmission, the energy consumption of the nodes and the end-to-end communication delay (Section 4.6).

Notations and symbols used in this chapter are summarized in Appendix 4.A.

4.2 Related work

Since the wireless nodes are usually energy-constrained, energy efficiency is a important requirement of WSN. To minimize the energy consumption of the nodes, various energy efficient mechanisms have been proposed in the literature. It has been observed that the energy consumption of a radio transceiver in the receive or transmit mode is much larger than the energy consumption of the sleep mode [30]. Thus, significant energy savings can be accomplished if the radio transceivers stay in sleep mode as long as possible. In [32], the authors have presented MAC scheduling mechanism (single hop communication), which reduces the number of the transitions between transmit, receive and sleep operation modes, and minimizes the time a wireless node needs to stay in high energy consuming modes (i.e. receive and transmit modes). The scheduling mechanism groups the operation modes on per node basis which improves the energy efficiency (extended sleep time), while reduces the bandwidth efficiency compared to the grouping on per operation mode basis. Sensor-MAC (S-MAC) [33] is a energy efficient MAC protocol for ad hoc WSNs based on the periodic listen/sleep schedule to reduce energy consumption. Neighbouring nodes are synchronized to go to sleep mode periodically. Adaptive listening reduces the increased end-to-end delay caused by nodes' periodic sleeping (when a sender gets a packet to transmit, it must wait until the receiver wakes up). A neighbouring node wakes up for a short time at the end of each transmission if it is a next-hop to which transmitter can pass data immediately. T-MAC [34] improves the energy efficiency of S-MAC by forcing all transmitting nodes to start transmission only at the beginning of each active period. B-MAC [35] has higher throughput and better energy efficiency than S-MAC and T-MAC. It adopts low power listening and clear channel sensing techniques to enhance channel utilization. In [33], the authors have also identified the major sources of energy waste in WSNs such as collisions, overhearing and idle listening. In this chapter, the above mentioned sources of the energy waste are eliminated by using the collision-free Time Division Cluster Scheduling and dedicated Guaranteed Time Slots (GTS) mechanisms.

There have been several research works dealing with the energy efficient routing protocols supporting QoS guarantees in WSNs. Real-time Power-Aware Routing (RPAR) protocol [36] integrates transmission power control and real-time routing for supporting energy efficient soft real-time communication in ad hoc WSNs. The protocol is based on the assumption that a higher transmission power results in higher speed. The transmission power is increased if the required speed is not satisfied, otherwise if the required speed is satisfied the transmission power is decreased (to improve energy efficiency). Another real-time routing algorithm minimizing the energy consumption was proposed in [37]. The authors have assumed a collision-free MAC protocol, and they have used multicommodity network flow model to schedule the optimal flows' paths in terms of energy consumption while not exceeding links' bandwidths and flows' deadlines. The routing algorithm ensures polynomial-time complexity but no scheduling is considered. Akkaya et al. [38] have proposed an energy-aware QoS routing protocol that find energy efficient path in a static cluster-based mesh WSN while meeting the end-to-end delay requirements of real-time traffic and maximizing the throughput of non real-time traffic. The authors assume that each node has a classifier to classify incoming real-time and non real-time traffic to different priority queues. The end-to-end delay requirements are converted into bandwidth requirements. However, their approach does not take into account the delay that occurs due to channel access at the MAC layer. Moreover, the use of priority queuing mechanism is too complex and costly for resource limited wireless nodes. On the contrary, this chapter assumes scheduling of cluster-tree WSNs where the flows' paths are unique, and the routing decisions are simple and time efficient.

The approach based on the combination of an energy efficient topology management protocol with a non energy-aware real-time routing protocol has been proposed in [39]. The nodes, which must be location-aware, are divided into the clusters. To reduce the energy consumption of the nodes while introducing a bounded delay, time-driven data transmissions within each cluster are performed in a Time Division Multiple Access (TDMA) fashion. The relay nodes, which use the real-time routing protocol to forward data between clusters towards the sink, are elected in rotation among the nodes belonging to each cluster. The Frequency Division Multiple Access (FDMA) mechanism prevents the collisions between nodes operating on different clusters. In [40], this topology management protocol has been extended to support even-driven data transmissions and dynamic cluster formation and reconfiguration.

To the best of our knowledge, so far no previous research has directly addressed the problem of energy efficient TDMA scheduling of time-bounded data flows in a cluster-tree WSN. Koubaa et al. [41] have proposed an algorithm for collision-free beacon/superframe scheduling in IEEE 802.15.4/ZigBee cluster-tree networks, using the time division approach. Note that the beacon frame scheduling problem comes back to a superframe scheduling problem, since each superframe starts with a beacon frame (Section 2.3). The authors have proposed an algorithm based on the ”pinwheel scheduling algorithm” [42], which performs the schedulability analysis of a set of superframes with different durations and beacon intervals, and provides a schedule if the set is schedulable. This problem becomes more complex and challenging when time-bounded data flows are assumed. Hence, this chapter addresses the problem of finding a collision-free superframe schedule that meets all data flows' end-to-end deadlines while minimizing the energy consumption of the nodes.

4.3 System model

This chapter considers a static deployment of wireless nodes which defines the physical topology of WSN given by the bidirectional wireless links between every pair of nodes that are within transmission range of each other. The logical topology, based on a physical topology, defines a subset of wireless links to be used for data transmission. In the rest of the thesis, the notation *topology* will be used while meaning logical topology.

4.3.1 Cluster-tree topology model

One of the WSN topologies suited for predictable and energy efficient behaviour is a cluster-tree (Figure 4.1) where the routing decisions are unique

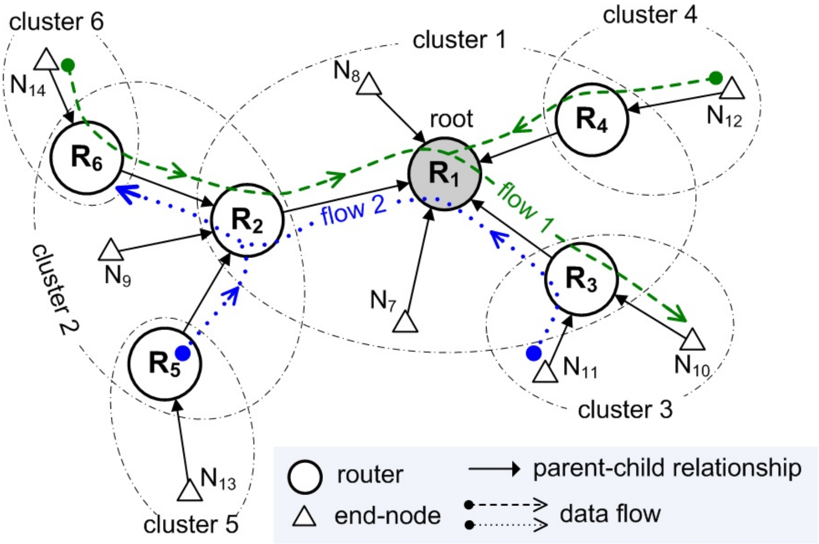


Figure 4.1: Cluster-tree topology with 2 time-bounded data flows.

and nodes can enter low power mode to save their energy. From the hierarchy point of view, the cluster-tree is directed tree (so called in-tree [43]) as depicted by solid arrows in Figure 4.1. On the other hand, from the data transmission point of view, the cluster-tree is undirected tree (i.e. the wireless links are bidirectional). The hierarchy of the cluster-tree topology is defined by parent-child relationships, in the sense that each solid arrow in Figure 4.1 leaves the *child* node and enters the *parent* node. Note that the in-tree has the following property: one node, called *root*, has no parent and any other node has exactly one parent.

The routers and end-nodes are two types of wireless nodes in cluster-tree WSNs. The nodes that can participate in multi-hop routing are referred to as *routers* (R_i). The nodes that do not allow association of other nodes and do not participate in routing are referred to as *end-nodes* (N_i). In the cluster-tree topology, the nodes are organized in logical groups, called *clusters*. Each router forms a cluster and is referred to as its *cluster-head* (e.g. router R_2 is the cluster-head of cluster 2). All of its child nodes (e.g. end-node N_9 and routers R_5 and R_6 are child nodes of router R_2) are associated to the cluster, and the cluster-head handles all their transmissions.

Throughout this thesis, the router and cluster-head are used interchangeably since each router R_i acts as a cluster-head of cluster i for all its child nodes, and as a consequence, will send periodic beacons to keep them synchronized.

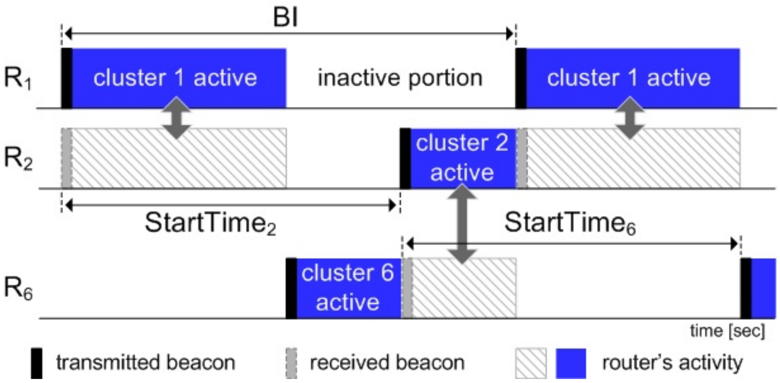


Figure 4.2: Timing among clusters 1,2 and 6 from Figure 4.1.

This *cluster-tree topology* (Figure 4.1) can be described by adjacency matrix $A = (a_{ij})$, where $a_{ij} = 1$ if router j is the parent router of node i, otherwise $a_{ij} = 0$. Remind [43] that A is a square matrix with dimension equal to the total number of nodes in a WSN (N_{node}^{TOTAL}).

In the cluster-tree topology, the multi-hop communication is deterministic because each node only interacts with its pre-defined parent router and child nodes. Messages are forwarded from cluster to cluster until reaching the sink. The time behaviour of each cluster is periodic and the period of each cluster is divided into two portions. *Active* portion (of duration SD), during which the cluster-head enables the data transmissions inside its cluster, and subsequent *inactive* portion. Each router (except the root) belongs to two clusters, once as a child node and once as a cluster-head. Hence, each router must be awake whenever one of these two clusters is active, otherwise it may enter the low-power mode to save energy (see the example in Figure 4.2). Router r has to maintain the timing between the active portion of its parent's cluster (in which a beacon and the data frames from the parent router are received, and the data frames to the parent router are sent) and its own active portion (in which a beacon and the data frames are sent to the associated child nodes, and the data frames from child nodes are received). Router r acts as a child node in the former active portion while in the latter active portion it acts as a cluster-head. The relative timing of these active portions is defined by the *StartTime* parameter [9]. The illustrative example is shown in Figure 4.2, where the router R_2 acts as child node in cluster 1 (shaded rectangle) and as cluster-head in cluster 2 (solid rectangle), for example.

Note that contrary to the balanced worst-case cluster-tree topology model considered in Chapter 5, this cluster-tree topology represents the

		flow 1	flow 2
sources		$\{N_{12}, N_{14}\}$	$\{R_5, N_{11}\}$
sink		N_{10}	R_6
e2e_deadline	[sec]	{0.1, 0.13}	{104, 135}
	[ptu]	{104, 135}	{52, 156}
req_period [sec]		0.4	1
sample_size [bit]		64	16
sample_ack		0	0

Table 4.1: The user-defined parameters of the data flows from Figure 4.1 (ptu = processing time unit).

configuration of practical WSN which can be balanced or unbalanced.

4.3.2 Data flow model

The traffic is organized in the data flows (see user-defined parameters of the flows from Figure 4.1 summarized in Table 4.1). Each data flow has one or more sources and exactly one sink. Both routers and end-nodes can have sensing or/and actuating capabilities, therefore, they can be sources or/and sinks of data flows. A node regularly measures a sensed value (e.g. temperature, pressure, humidity) with the required period, called the *req_period*, and reports the acquired sensory data of a given size, called the *sample_size*, to a sink. Note that *req_period* defines the minimal inter-arrival time between two consecutive measurements, and a particular inter-arrival time has to be greater or equal to the *req_period*.

End-to-end (e2e) delay d_{ij}, given as a time between the instant when a source i sends the message and the instant when the sink j receives this message, is bounded by $e2e\_deadline_{ij}$ such that $d_{ij} \leq e2e\_deadline_{ij}$. Note that this parameter is set for each source of a particular data flow, and all of them must be met.

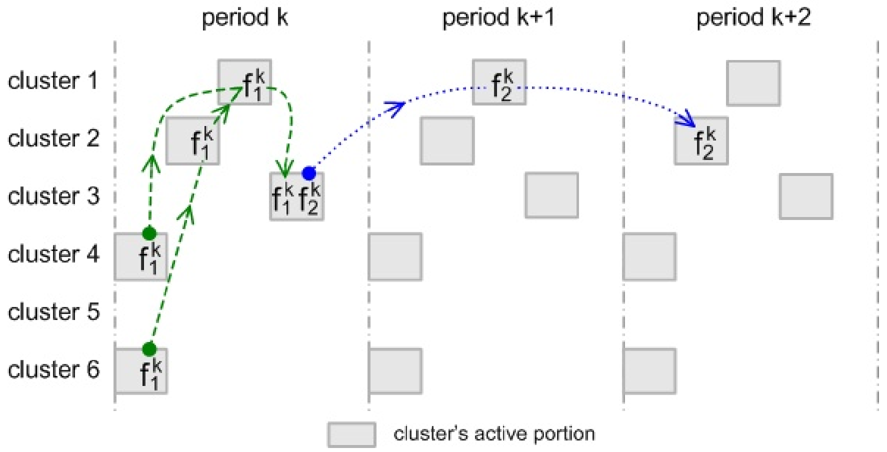
The communication errors such as message corruption or message loss come from unreliable and time-varying characteristics of wireless channels [8]. A corrupted or lost message can be detected by the simple checksum or acknowledgement techniques, respectively and restored by the retransmission mechanism, for example. All of the above mentioned mechanisms are natively supported by the IEEE 802.15.4 protocol [9]. The messages of a given data flow can be transmitted without acknowledgement, i.e. parameter *sample_ack* = 0, or with acknowledgement, i.e. *sample_ack* = 1. Note that the maximum number of retransmissions must be bounded, otherwise, the analysis will not be possible.

The length of active portion, and consequently the length of schedule, grows with the maximum number of retransmissions. Given a channel error rate, the simulation study is used to show interdependence of timeliness, energy consumption and reliability, in a way that improving one may degrade the others (Section 4.6).

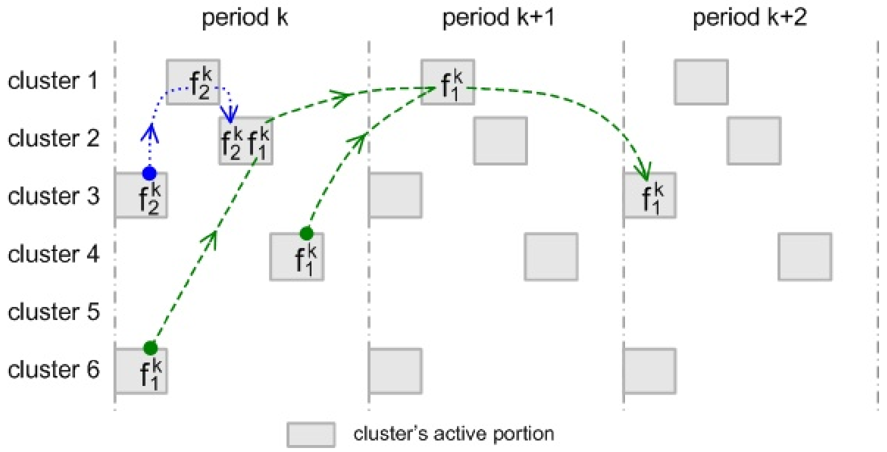
4.3.3 Cyclic nature

In cluster-tree WSNs, the flows traverse different clusters on their routing paths from the source nodes to the sink nodes. One execution of the flow (i.e. complete data communication from the source node/nodes to the sink node) is called a *wave*, and the notation f_i^k is used to denote wave k of the flow i. The flows are assumed to be transmitted with the same period, therefore wave f_i^k is followed by wave f_i^{k+1} for all flows and all waves with the same time separation. The cluster is active only once during the period [10], therefore all the flows in a given cluster are bound together. For example, the gray rectangles on the first line of Figure 4.3 show active portions of cluster 1 during three consecutive periods accommodating flows 1 and 2 in each period. The key problem is to find a periodic schedule, called *Time Division Cluster Schedule* (TDCS), which specifies when the clusters are active while avoiding possible inter-cluster collisions and meeting all data flows' e2e deadlines. The schedule is characterized not only by the moments when the clusters become active within the period, but due to the cyclic nature of the problem it is also characterized by the index of the wave for each flow in a given cluster.

Figure 4.3 shows two possible schedules of the example in Figure 4.1. Even if we relax on the lengths of transmitted messages and on resource constraints related to the cluster collisions, we have to deal with the precedence relations of the wave traversing different clusters. Since the flows have opposite directions in this example, the e2e delay minimization of the first flow is in contradiction with the the minimization of the second flow. Figure 4.3a shows the case, when e2e delay of the flow 1 is minimized, i.e. the ordered sequence of clusters' active portions is in line with the flow 1 (starting with clusters 4 and 6 and following with clusters 2, 1 and 3), and therefore one wave of this flow fits into one period. On the other hand, the wave of the flow 2 spans over 3 periods while going against the sequence of clusters.



(a) Minimized end-to-end delay of flow 1 (dashed line)



(b) Minimized end-to-end delay of flow 2 (dotted line)

Figure 4.3: Schedules for data flows in Figure 4.1.

Figure 4.3b illustrates the opposite case, when e2e delay of the flow 2 is minimized (starting with cluster 3 and following with clusters 1 and 2), and consequently flow 1 spans over 3 periods. It may happen that none of these schedules is feasible due to the deadline constraints (even if feasible schedule exists - see Figure 4.8). Hence, proper order of the active portions of clusters is a subject of optimization even if the lengths of messages and collisions of clusters are not assumed.

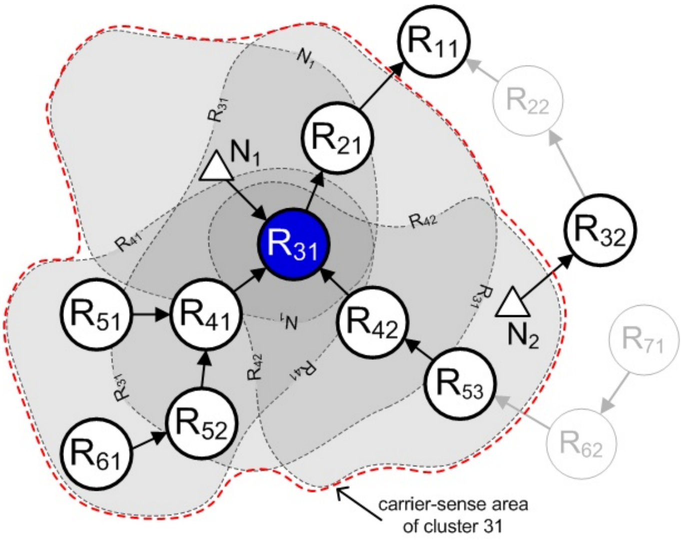


Figure 4.4: The carrier-sense area and collision domain (bold routers) of cluster 31.

4.3.4 Collision domains

Each wireless node is equipped with a radio transceiver together with an antenna. According to the strength of the radio signal, the transmission range and the carrier-sensing range [44] can be defined around each transmitter. When a receiver is in the *transmission range* of a transmitter, it can receive and correctly decode messages from the transmitter. On the other hand, a node in the *carrier-sensing range* (also called the hearing range), but not in the transmission range, is able to sense the transmission (or even significant radio energy), but cannot decode the messages correctly. The carrier-sensing range is always larger than the corresponding transmission range [44]. However, both ranges depend on the transmit power level, the parameters of a given antenna and the surroundings' conditions. In what follows, the topology is given by the transmission ranges (i.e. each node must be within the transmission range of at least one other node) while the collision domains depend on the carrier-sense ranges.

A *carrier-sense area* of a cluster is covered by the overlapping carrier-sense ranges of its cluster-head and associated child nodes. A *collision domain* of a cluster is a set of clusters, which compete for the same radio channel and, therefore, their active portions must be non-overlapping, i.e. only one cluster from a collision domain can be active at a given time instant. The collision domain of cluster i comprises the cluster j if and only if the carrier-sense area of cluster i comprises cluster-head or any of child nodes of cluster j. Hence the collision domain depends on the physical deployment of a WSN as well as on the topology (i.e. parent-child relationships). The collision domain is defined for each cluster in a WSN.

Let us consider the example in Figure 4.4. The carrier-sense area of cluster 31 (gray region in Figure 4.4) is covered by the carrier-sense ranges of cluster-head R_{31} and its child nodes (i.e. routers R_{41}, R_{42} and end-node N_1). Hence, the collision domain of cluster 31 comprises the clusters 31, 41, 42, 51, 52, 53, 61, 21 whose cluster-heads are inside the carrier-sense area (i.e. R_{31}, R_{41}, R_{42}, R_{51}, R_{52}, R_{53}, R_{61}, R_{21}), and the clusters 11 and 32 whose child nodes are inside the carrier-sense area (i.e. R_{21} and N_2). On the other hand, the collision domain of cluster 31 does not comprise the clusters 22, 62, 71.

The collision domains of a WSN are defined by collision matrix $C = (c_{ij})$, where $c_{ij} = 1$ if cluster j is within the collision domain of cluster i, otherwise $c_{ij} = 0$. Then, C is a square matrix with dimension equal to the total number of clusters in a WSN (N_{router}^{TOTAL}).

4.4 Time Division Cluster Scheduling and its application to IEEE 802.15.4/ZigBee

The messages are being passed through the cluster-tree WSN from cluster to cluster until reaching their sink. To avoid inter-cluster collisions (messages/beacons transmitted from nodes in different overlapping clusters), it is mandatory to schedule the clusters active portions in an ordered sequence that is called *Time Division Cluster Schedule* (TDCS). It is easy to see that in a network with multiple collision domains, the clusters from different non-overlapping collision domains may be active at the same time (i.e. some clusters' active portions can run simultaneously).

The TDCS significantly affects the resource requirements and delay bounds in cluster-tree WSNs. The objective of this chapter is to minimize the energy consumption of the nodes by maximizing the TDCS period, corresponding to BI, while avoiding possible inter-cluster collisions (i.e. resource requirements) and meeting all data flows' end-to-end deadlines (i.e. temporal requirements). Note that to minimize the energy consumption of nodes, the lowest duty-cycles must be chosen (IEEE 802.15.4 supports duty-cycles under 1%). All clusters have equal BI, defined by BO, but various SD (Section 4.4.1), defined by SO, (i.e. various duty-cycle) to ensure efficient bandwidth utilization. The BI should be set as long as possible to minimize clusters' duty-cycle and, consequently, to minimize the energy consumption of the nodes. As a result, the clusters inactive portion is extended, and the nodes may stay in the low power mode longer to save energy. On the other hand, minimum duty-cycles enlarge the end-to-end delays. Hence, long lifetime is in contrast to the fast timing response of a WSN, therefore the interest in in finding the TDCS minimazing the duty-cycles while respecting all of the required data flows e2e deadlines.

The key idea of this chapter is to formulate the problem of finding a feasible TDCS as a cyclic extension of the RCPS/TC (Resource Constrained Project Scheduling with Temporal Constraints) problem [45], so that the users are not restricted to a particular implementation but they can make a similar extension to any of the algorithms solving this problem.

The TDCS algorithm (formulated in Sections 4.4.2, 4.4.3 and solved in Section 4.4.4) is called iteratively starting from the minimum BI up to the maximum BI. The maximum BI, given by BO_{max} in Eq.(2.1), is equal to or shorter than the shortest *req_period* among all of the data flows. The minimum BI, given by BO_{min}, is equal to or longer than the duration of all clusters' SDs when assuming that non-interfering clusters overlap. If a feasible TDCS is found for a given BI, BO is increased by 1 and the TDCS algorithm is called once again with new BI. This procedure is repeated until $BO = BO_{max}$ or a feasible TDCS is not found. Then, the last feasible TDCS meets all the resource and temporal requirements while minimizing the energy consumption of the nodes.

4.4.1 Duration of the cluster's active portion in IEEE 802.15.4/ZigBee

The duration of the cluster's active portion (SD) is given by the amount of data traffic traversed through a given cluster. To reduce the resource requirements of the routers and end-to-end delays, the following priority rule is introduced: "*When a cluster-head handles several GTSs in opposite directions, the transmit GTSs (i.e. communication from child-node to cluster-head) are allocated before the receive GTSs (i.e. communication from cluster-head to child-node)*". Using this rule, the end-to-end delay of a flow can be reduced by one period of TDCS at a router which allocates both receive and transmit GTSs for a given flow. For example, in Figure 4.1, router R_1 allocates one transmit GTSs and one receive GTS for flow 1. In order to reduce the computational complexity, the GTSs inside the transmit or receive group are in an arbitrary order and are not the subject of the optimization.

The length of each GTS is given by the amount of transmitted data. Each GTS includes effective data transmission and overheads (i.e. inter-frame spacing (IFS) and eventual acknowledgement and retransmissions - see Section 2.3). The whole transmission, including the data frame, IFS and eventual acknowledgement and retransmissions, must be completed before the end of the current GTS. Otherwise, it must wait until the GTS in the next superframe.

The duration of a GTS required for the whole data transmission is expressed as:

$$T_{GTS} = \sum_{i=1}^{e} \left(\begin{array}{c} \Delta IFS_i + (macMaxFrameRetries \cdot sample\_ack_i + 1) \cdot \\ (frm\_size_i / rate + macAckWaitDuration \cdot sample\_ack_i) \end{array} \right) \tag{4.1}$$

where frm_size is the size of transmitted frame including the data payload, MAC and PHY headers; $rate$ is the data rate equal to 250 kbps; ΔIFS is equal to SIFS or LIFS depending on the length of MAC frame; and e is the number of 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:

$$N_{GTS} = \left\lceil \frac{T_{GTS}}{TS} \right\rceil \tag{4.2}$$

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 superframe. The remaining time slots of 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* [9].

The superframe duration (SD) is then computed iteratively starting from $SO = 0$. If the number of time slots required for all allocated GTSs in a given superframe is greater than $16 - \lceil aMinCAPLength/TS \rceil$, the SO is increased by 1 and the length of each GTS (Eq. (4.2)) is recalculated. This procedure is repeated until all allocated GTSs fit into a given SD.

To ensure efficient bandwidth utilization, the SD of the clusters handling a higher amount of data traffic should be longer than the ones handling less amount of data traffic. Thus, the adequate SD is computed for each cluster such that for each cluster k, we get SO_k and the configuration parameters [9] of each allocated GTS, i.e. *GTS device*, *GTS direction*, *GTS length* and *GTS starting slot*. In case of the example in Figure 4.1, the configuration parameters are summarized in Table 4.2. Note that all clusters have the same

BO equal to 3, which gives the longest possible BI minimizing the energy consumption of the nodes.

cluster	SO	GTS device	GTS length	GTS direction	GTS starting slot
cluster 1	1	R_2	1	transmit	10
		R_3	1	transmit	11
		R_4	1	transmit	12
		R_2	1	receive	13
		R_3	2	receive	14
cluster 2	0	R_5	2	transmit	8
		R_6	2	transmit	10
		R_6	4	receive	12
cluster 3	0	N_{11}	2	transmit	10
		N_{10}	4	receive	12
cluster 4	0	N_{12}	2	transmit	14
cluster 6	0	N_{14}	2	transmit	14

Table 4.2: Guaranteed bandwidth of clusters' superframes corresponding to the example in Figure 4.1.

4.4.2 TDCS formulated as a cyclic extension of RCPS/TC

The concept of *(general) temporal constraints* (also called *minimum and maximum time lags*) have been classified by Brucker et al. [46]. The problem was studied by the operations research community, but similar principles have also appeared in the optimization of compilers for multiprocessor machines [47] and in symbolic representation of states in timed automata [48].

The set of n tasks $\mathcal{T} = \{T_1, \ldots T_i, \ldots T_n\}$ with temporal constraints is given by a graph of communication tasks G (see Figure 4.6), where the vertices correspond to the tasks and the directed edges represent the temporal constraints between the tasks. The scheduling problem is then defined as searching for such a *feasible schedule* $(s_1, s_2, \ldots s_n)$, which satisfies the *temporal constraints* and *resource constraints* while minimizing the objective criterion. Note that in RCPS/TC terminology, s_i is the start time of task T_i related to the beginning of the schedule (i.e. time 0), but in IEEE 802.15.4/ZigBee terminology, the parameter *StartTime*, is related to the moment, when the beacon frame from the parent router was received (Figure 4.2). Parameter *StartTime* can be easily derived from start time

s and matrix A, since parameter $StartTime$ of the root is equal to 0 (see Eq. (4.12) for more details).

Each edge from vertex T_i to vertex T_j is labelled by a weight $w_{i,j}$ which constraints the start times of the tasks T_i and T_j by the inequality $s_j - s_i \geq w_{i,j}$. There are two kinds of edges: the edges with positive weights and the edges with negative weights. The edge, from vertex T_i to vertex T_j with a positive weight $w_{i,j}$ (giving the minimum time lag), indicates that s_j, the start time of T_j, must be at least $w_{i,j}$ time units after s_i, the start time of T_i (i.e. $s_j \geq s_i + w_{i,j}$). We use the positive weights $w_{i,j}$ to represent the *precedence constraint*, i.e. $w_{i,j} = p_i$ and therefore T_j starts after completion of T_i, where p_i is processing time of T_i.

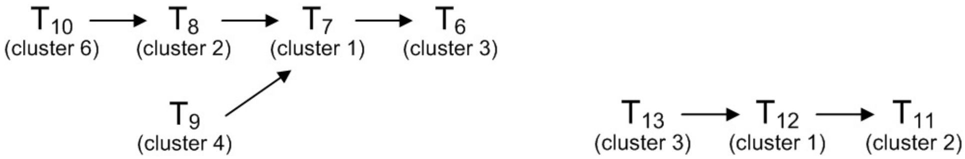
The edge, from vertex T_j to vertex T_i with a negative weight $w_{j,i}$ (giving the maximum time lag), indicates that s_j must be no more than $|w_{j,i}|$ time units after s_i (i.e. $s_j \leq s_i + w_{j,i}$). Therefore, each negative weight $w_{j,i}$ represents the *relative deadline* of task T_j in relation to task T_i. Consequently, when T_j is the last task of the flow and T_i is the first task of the same flow, the edge with a negative weight may be conveniently applied for e2e deadline such that the value of e2e deadline is equal to $|w_{j,i}| + p_j$.

Any feasible TDCS has to respect the resource constraints related to the collision domains of clusters and the temporal constraints of the flows. Hence, the set of tasks $\mathcal{T}$ will consist of two disjoint subsets: a set of cluster-tasks and a set of dummy-tasks reflecting the temporal constraints only. The duration of a task T_i is given by its processing time p_i.

The *cluster-task* T_i is created for each cluster i. Note that the clusters which do not route any data flow have $p_i = 0$ (i.e. cluster-task 5 is not shown in Figure 4.6). In the case of an active cluster-task (i.e. the one routing at least one data flow) the processing time is equal to the cluster's SD (i.e. p_i = SD computed in Section 4.4.1) and includes all the communications handled by the given cluster. That means, for each cluster-task, we define the duration of the CAP as p_i^{CAP}, the duration of all GTSs in the transmit direction as p_i^T and the duration of all GTSs in the receive direction as p_i^R, i.e. the processing time of cluster-task is given as $p_i = p_i^{CAP} + p_i^T + p_i^R$. The unit of processing time, called *ptu*, is equal to the length of a time slot when $SO = 0$ (i.e. 1 ptu = $aBaseSuperframeDuration/16 = 0.96$ ms).

Each dummy-task has a processing time equal to 0 since they are used to handle temporal constraints of different flows.

Let us consider the illustrative example of cluster-tree WSN in Figure 4.1, where periodic time-bounded traffic is sent using two data flows. Within the first data flow, messages are sent from source nodes N_{12} and N_{14} to the



(a) In-tree of data flow 1. (b) In-tree of data flow 2.

Figure 4.5: In-trees of dummy-tasks relating to the data flows.

sink node N_{10}. In the second case, nodes R_5 and N_{11} send messages to the sink router R_6. The user-defined parameters of the flows are summarized in Table 4.1. Thus, cluster-tasks T_1, T_2, T_3, T_4, T_5, T_6 are associated with clusters 1, 2, 3, 4, 5 and 6. The processing time of each cluster-task is equal to its associated cluster's SD as follows: $p^{CAP} = [20, 8, 10, 14, 0, 14]$, $p^T = [6, 4, 2, 2, 0, 2]$ and $p^R = [6, 4, 4, 0, 0, 0]$. Cluster 5 does not route any flow, thus its processing time is equal to 0. Note that since Superframe Order of cluster 1 is equal to 1 (Table 4.2), the processing time p_1 was doubled.

The collisions among the routers are represented as conflicts among the cluster-tasks due to the shared resources. $\mathcal{M}$ is defined as a set of couples of the cluster-tasks having a potential conflict. Consider two cluster-tasks T_i and T_j. The potential conflict between T_i and T_j is a couple $\{i, j\}$ derived from the collision matrix C as follows: we say that $\{i, j\} \in \mathcal{M}$ if and only if $C_{i,j} = 1$, $p_i > 0$ and $p_j > 0$.

Precedence constraints of each flow are represented by an in-tree of dummy-tasks connected by positive edges. The leaves correspond to the source clusters, where the source nodes are associated, and the root to the sink cluster, where the sink is associated. In particular example in Figure 4.6, the in-tree of dummy-tasks T_{11}, T_{10}, T_9, T_8, T_7 corresponds to flow 1, and the in-tree of dummy-tasks T_{14}, T_{13}, T_{12} corresponds to flow 2. Each dummy-task represents a given flow in a given cluster (e.g. T_{11} represents flow 1 in cluster 6).

Precedence constraints of each flow are represented by an in-tree of dummy-tasks connected by positive edges. The leaves correspond to the source clusters, where the source nodes are associated, and the root to the sink cluster, where the sink is associated. In particular example in Figure 4.1, the in-tree of dummy-tasks T_{11}, T_{10}, T_9, T_8, T_7 corresponds to flow 1 (Figure 4.5a), and the in-tree of dummy-tasks T_{14}, T_{13}, T_{12} corresponds to flow 2 (Figure 4.5b). Each dummy-task represents a given flow in a given cluster (e.g. T_{11} represents flow 1 in cluster 6).

The messages transmitted periodically over the network can be considered as a periodic execution of task-set $\mathcal{T}$. As mentioned in Section 4.3.3, one wave of a given flow may go over several periods and, therefore, we are faced with a cyclic scheduling problem.

Let $\hat{s}_i$ be the start time within the period, i.e. remainder after division of s_i by BI, and let $\hat{q}_i$ be the index of the period, i.e. the integer part of this division. Then start time s_i can be expressed as follows:

$$s_i = \hat{s}_i + \hat{q}_i \cdot \mathrm{BI} \qquad \text{for} \quad \hat{s}_i \in \langle 0, \mathrm{BI}-1 \rangle\,,\ \hat{q}_i \geq 0. \tag{4.3}$$

This notation divides s_i into segment $\hat{q}_i$ and offset $\hat{s}_i$. Hence, two tasks T_i and T_j within one period may have a different $\hat{q}_i$ and $\hat{q}_j$, since the pieces of data related to these tasks correspond to the different waves (this notion used in cyclic scheduling is identical to the modulo scheduling or SW pipelining in the parallel compiler community [49]).

The cyclic schedule has to follow several constraints:

- *Precedence constraints and relative deadlines* are given by inequality $s_j - s_i \geq w_{i,j}$. As a result, by applying Eq. (4.3) we obtain:

  $$(\hat{s}_j + \hat{q}_j \cdot \mathrm{BI}) - (\hat{s}_i + \hat{q}_i \cdot \mathrm{BI}) \geq w_{i,j}. \tag{4.4}$$

- *Offset precedence constraints and offset relative deadlines* are used to bind the flow-related dummy tasks with the cluster-task. They represent the relation between two tasks that can be from different waves. Therefore, they do not contain the segment values $\hat{q}$ and can be expressed as:

  $$\hat{s}_j - \hat{s}_i \geq v_{i,j}. \tag{4.5}$$

  The offset weights $v_{i,j}$ are used to distinguish the *offset precedence constraints* from *”normal” precedence constraints.*

- *Resource constraints* given by $\mathcal{M}$, the set of potential conflicts of the cluster-tasks. The conflicts have to be avoided in order to obtain a feasible schedule (detailed explanation is given in Section 4.4.4).

4.4.3 Graph of the communication tasks

An important step of the scheduling algorithm is the construction of the graph of the communication tasks G (Figure 4.6) using the data flows in Table 4.1 and topology in Figure 4.1 (i.e. adjacency matrix A and collision matrix C).

Each dummy-task is synchronized with the corresponding cluster-task. The synchronization is made by means of *offset precedence constraints*

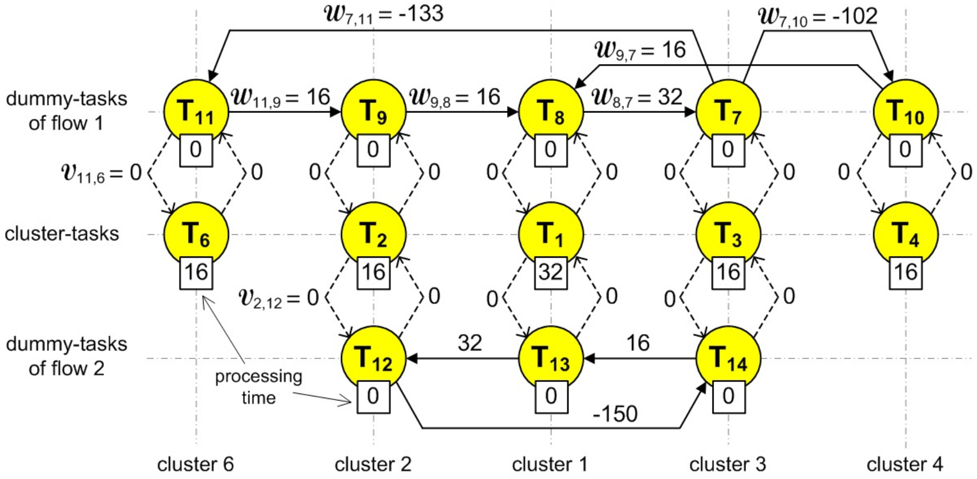


Figure 4.6: Graph G of the tasks corresponding to example in Figure 4.1.

represented by dashed edges in Figure 4.6. All of them have the weight $v_{i,j} = 0$, therefore, for example, $\hat{s}_6 = \hat{s}_{11}$ is given by two inequalities (4.5), i.e. $\hat{s}_6 \geq \hat{s}_{11}$ and $\hat{s}_6 \leq \hat{s}_{11}$.

Positive edges are used to represent precedence constraints of the flows. For example of flow 1, dummy-task T_9 starts after dummy-task T_{11} is completed, which is represented by the positive edge with weight $w_{11,9}$ equal to the processing time of cluster-task T_6, i.e. $w_{11,9} = p_6 = 16$.

Negative edges are used to represent the e2e deadlines of the flows. The e2e deadline of the flow spans from the beginning of transmit or receive GTS's groups to the end of transmit or receive GTS's groups (see Figure 4.9). On the other hand, the relative deadline between corresponding tasks, given by the weight $w_{i,j}$, starts and ends at the beginning of the tasks. Hence, the e2e deadline must be aligned with the beginning of corresponding tasks. For example of a sub-flow of flow 1 from source end-node N_{14} to sink end-node N_{10}, the relative deadline between the corresponding dummy-tasks T_7 and T_{11} is given by the weight $w_{7,11}$ as follows:

$$\begin{aligned} w_{7,11} &= -\big(e2e\_deadline_{N_{14}N_{10}} + (p_6^{CAP} + p_6^T \cdot \theta_{1,6}) \\ &\qquad\qquad - (p_3^{CAP} + p_3^T + p_3^R \cdot (1 - \theta_{1,3}))\big) \\ &= -(135 + (14 + 2 \cdot 0) - (10 + 2 + 4 \cdot 1)) = -133 \text{ ptu} \end{aligned} \tag{4.6}$$

where $\theta_{f,r}$ is a binary constant, which is equal to 1 when router r is source/sink of flow f and is equal to 0 when a child node of router r is

$$\min \sum_{i=1}^{n} \hat{s}_i + \hat{q}_i \cdot \mathrm{BI} \tag{4.7}$$

subject to:

$$\hat{s}_j + \mathrm{BI} \cdot \hat{q}_j - \hat{s}_i - \mathrm{BI} \cdot \hat{q}_i \geq w_{ij} \qquad \forall (i,j); i \neq j, w_{ij} \neq -\infty \tag{4.8}$$

$$\hat{s}_j - \hat{s}_i \geq v_{ij} \qquad \forall (i,j); i \neq j, v_{ij} \neq -\infty \tag{4.9}$$

$$\hat{s}_i - \hat{s}_j + \mathrm{BI} \cdot x_{ij} \geq p_j \qquad \forall \{i,j\} \in \mathcal{M}; i < j \tag{4.10}$$

$$\hat{s}_i - \hat{s}_j + \mathrm{BI} \cdot x_{ij} \leq \mathrm{BI} - p_i \qquad \forall \{i,j\} \in \mathcal{M}; i < j \tag{4.11}$$

where: $\hat{s}_i \in \langle 0, \mathrm{BI} - p_i \rangle$; $\hat{q}_i \geq 0$; $\hat{s}_i, \hat{q}_i \in \mathbb{Z}$; $x_i \in \{0,1\}$

Figure 4.7: ILP formulation for cyclic extension of the scheduling problem.

source/sink of flow f. The resulting end-to-end delay ($d_{N_{14}N_{10}}$) is constrained by $e2e\_deadline_{N_{14}N_{10}}$ (see Figure 4.9), and it spans from the beginning of transmit GTS's group of cluster 6 (since the measured data has to be received from the end-node N_{14} through a transmit GTS) to the end of receive GTS's group of cluster 3 (since the received data has to be dispatched to the end-node N_{10} through a receive GTS).

Graph G in Figure 4.6 is given by W, the adjacency matrix of the weights $w_{i,j}$, and V, the adjacency matrix of the offset weights $v_{i,j}$. If there is no edge from T_i to T_j, then $w_{i,j} = v_{i,j} = -\infty$.

4.4.4 Solution of the scheduling problem by integer linear programming algorithm

In this part, an Integer Linear Programming (ILP) formulation for cyclic extension of RCPS/TC (Figure 4.7) is defined. Let x_{ij} be a binary decision variable such that $x_{ij} = 1$ if and only if $\hat{s}_i \leq \hat{s}_j$ (i.e. T_i is followed by T_j or both T_i and T_j start at the same time) and $x_{ij} = 0$ if and only if $\hat{s}_i > \hat{s}_j$ (i.e. T_j is followed by T_i).

Note that the period BI, vector p, matrices W, V and the set of potential conflicts $\mathcal{M}$ are input parameters of the declarative program in Figure 4.7.

Constraint (4.8) is a direct application of the precedence constraints and relative deadlines given by W. Constraint (4.9) relates to the offset precedence constraints and offset relative deadlines given by V. Constraints (4.10) and (4.11) limit the number of tasks executed at a given time. The binary decision variable x_{ij} defines the mutual relation of tasks T_i and T_j ($i \neq j$) within the period as follows:

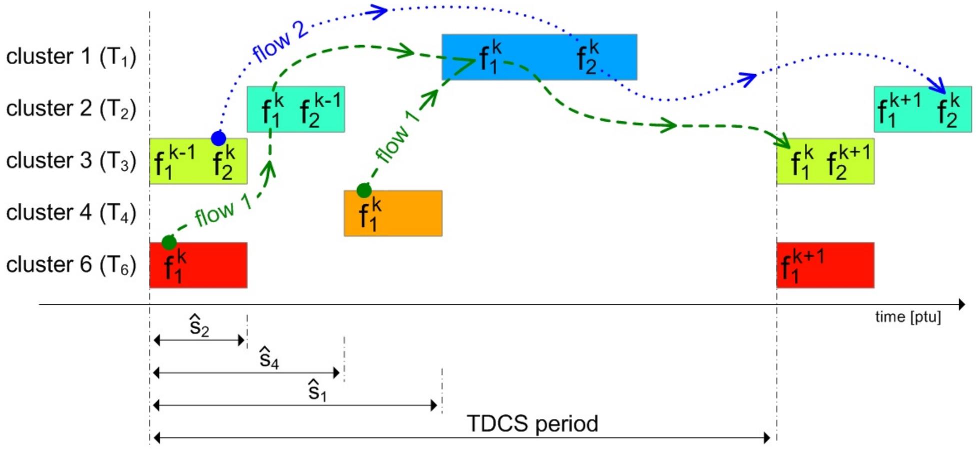


Figure 4.8: Gantt chart of TDCS including flows 1 and 2.

1. When $x_{ij} = 0$, constraint (4.11) is eliminated in effect (since $\hat{s}_i - \hat{s}_j + \mathrm{BI} \geq p_j$ is always true with respect to the definition domain of variable s) and constraint (4.10) reduces to $\hat{s}_j + p_j \leq \hat{s}_i$, i.e. T_j is followed by T_i within the period.
2. When $x_{ij} = 1$, constraint (4.10) is eliminated in effect and constraint (4.11) reduces to $\hat{s}_i + p_i \leq \hat{s}_j$, i.e. T_i is followed by T_j within the period.

The above mentioned scheduling algorithm have been implemented in the Matlab [50] using GLPK solver [51]. Figure 4.8 shows the offsets of start times ($\hat{s}$) of cluster-tasks (namely $\hat{s}_1 = 48$, $\hat{s}_2 = 16$, $\hat{s}_3 = 0$, $\hat{s}_4 = 32$, $\hat{s}_6 = 0$) in the form of a Gantt chart for one whole period of a feasible TDCS of the cluster-tree WSN in Figure 4.1 including flows 1 and 2 along the wave k. The value of start times $\hat{s}$ is in processing time units (ptu). Note that the cluster-tasks T_3 and T_6 can overlap because the collision domain of cluster 3 does not include cluster 6 and vice versa. In addition, the output of algorithm contains the index ($\hat{q}$) of the TDCS period for each flow related to dummy-task as follows: $\hat{q}_{11} = 0$, $\hat{q}_{10} = 0$, $\hat{q}_9 = 0$, $\hat{q}_8 = 0$, $\hat{q}_7 = 1$ and $\hat{q}_{14} = 0$, $\hat{q}_{13} = 0$, $\hat{q}_{12} = 1$.

The *StartTime* parameter of each cluster's active portion (except the root) is computed from the offset of start times as follows:

$$StartTime_i = \hat{s}_i + \gamma \cdot \mathrm{BI} - \hat{s}_{parent} \tag{4.12}$$

where $\hat{s}_{parent}$ is the offset of start time of the parent cluster-task of cluster-task i, and $\gamma = 1$ if $\hat{s}_i < \hat{s}_{parent}$; otherwise $\gamma = 0$. The *StartTime* parameter of the active portion of the root is equal to 0. In case of the example in Figure 4.1 in which $BO = 3$ (BI = 128 ptu), the *StartTime* parameter of the active portion of cluster 4 is then computed as: $StartTime_4 = \hat{s}_4 + 1 \cdot \mathrm{BI} - \hat{s}_1 = 32 + 128 - 48 = 112$ ptu.

Using Eq. (4.3), the e2e delay between each source and sink of a given flow is computed. For example of a sub-flow of flow 1 from source end-node N_{14} to sink end-node N_{10}, the e2e delay is computed as follows:

$$\begin{aligned} d_{N_{14}N_{10}} &= \left(s_7 + p_3^{CAP} + p_3^{R} + p_3^{T} \cdot (1 - \theta_{1,3})\right) - \left(s_{11} + p_6^{CAP} + p_6^{T} \cdot \theta_{1,6}\right) \\ &= \left((128 + 10 + 2 + 4 \cdot (1 - 0)\right) - \left(0 + 14 + 2 \cdot 0\right) = 130 \text{ ptu} \end{aligned} \tag{4.13}$$

where binary constant $\theta_{f,r}$ has been defined in Eq. (4.6); s_7 and s_{11} are start times of corresponding dummy-tasks T_7 and T_{11}; p_3 and p_6 are processing times of cluster 3 and 6, respectively, where end-nodes N_{10} and N_{14} are associated.

The time line of clusters' active portions including allocated GTSs is presented in Figure 4.9. The *StartTime* parameters and e2e delays are based on the values of $\hat{s}$, $\hat{q}$ and using of Egs. (4.13) and (4.12).

Using the proposed scheduling methodology, system designers are able to configure the parameters of each cluster, such as *BO*, *SO* and *StartTime*, in IEEE 802.15.4/ZigBee cluster-tree WSNs. Furthermore, for every cluster's superframe, the configuration parameters [9] of each allocated GTS such as *GTS device*, *GTS direction*, *GTS length* and *GTS starting slot* can be obtained as well (Table 4.2).

4.5 Time complexity

This section focuses on the time complexity of the proposed TDCS algorithm implemented in Matlab while using the simplex-based GLPK solver [51] (GNU Linear Programming Kit by A. Makhorin). The time complexity usually depends on the number of decision variables, which is in this case less than $(n_c^2 - n_c)/2 + n_d$, where n_c stands for the number of active cluster-tasks (x_{ij} is generated for each couple of potentially conflicting tasks) and n_d stands for the number of dummy-tasks ($\hat{q}_i$ is generated for each of them).

In this experiment, the system model is configured as follows. The number of child routers is randomly generated for each parent router and varies between 0 and 3. The routers are successively generated until the

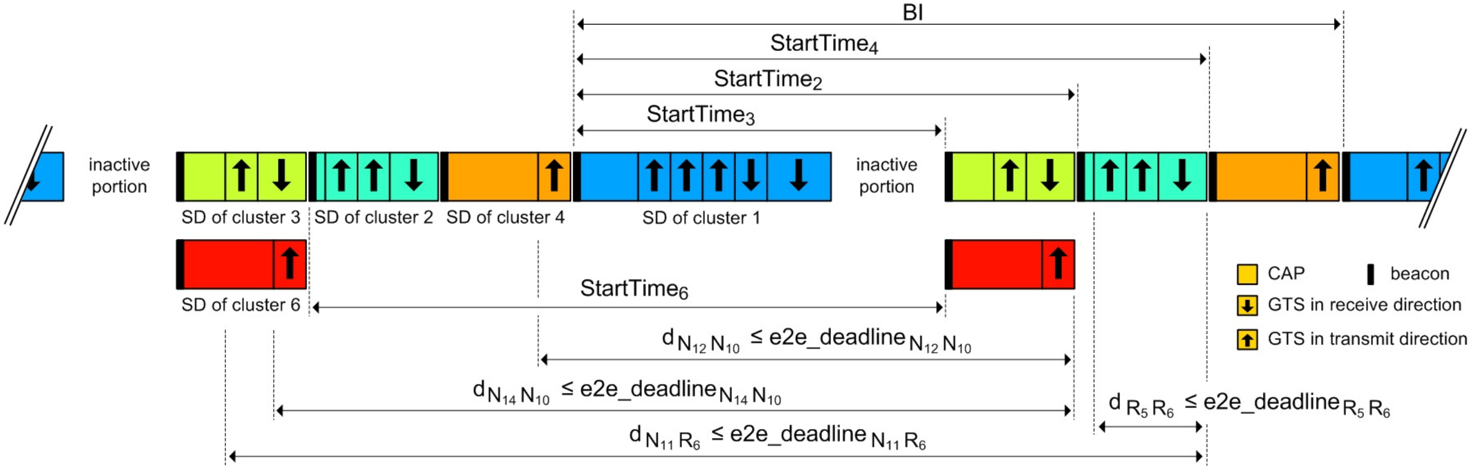

BI
StartTime$_4$
StartTime$_2$
StartTime$_3$
inactive portion
inactive portion
SD of cluster 3
SD of cluster 2
SD of cluster 4
SD of cluster 1
SD of cluster 6
StartTime$_6$
$d_{N_{12} N_{10}} \leq e2e\_deadline_{N_{12} N_{10}}$
$d_{N_{14} N_{10}} \leq e2e\_deadline_{N_{14} N_{10}}$
$d_{N_{11} R_6} \leq e2e\_deadline_{N_{11} R_6}$
$d_{R_5 R_6} \leq e2e\_deadline_{R_5 R_6}$
CAP
beacon
GTS in receive direction
GTS in transmit direction

Figure 4.9: Time line of TDCS corresponding to the example in Figure 4.1.

N_{router}^{TOTAL} (N_{node}^{TOTAL})	N_{flow}	N_{source}	N_{task}	$time_{compact}$ [sec]	$time_{feasible}$ [sec]
7 (28)	2	3	15.35	0.106	0
		6	14.75	0.107	0
	4	3	24.55	0.184	0
		6	24.15	0.228	0
11 (44)	2	3	18.25	0.170	0.159
		6	18.9	0.274	0.1
	4	3	31.35	0.547	0.263
		6	33.05	1.002	0.231
16 (64)	2	3	25.61	2.166	0.244
		6	26.91	9.971 (1)	0.351
	4	3	46.1	21.124 (2)	5.359
		6	48.05	25.88 (2)	5.648
20 (80)	2	3	25.35	6.7588 (2)	0.243
		6	24.77	56.3649 (2)	0.350
	4	3	44.43	–	7.778
		6	43.75	–	10.809
40 (160)	2	3	36.65	–	17.649
		6	34.90	–	21.90
	4	3	66.30	–	100.2
		6	68.95	–	106.84
60 (240)	2	3	40.9	–	19.00
		6	40.15	–	29.93 (2)
	4	3	73.75	–	142.89 (2)
		6	75.4	–	158.65 (4)

Table 4.3: Time complexity of TDCS algorithm.

total number of routers in the network reaches N_{router}^{TOTAL}. Each router has 3 child end-nodes. Note that the locations of child routers and end-nodes are randomly generated within the transmission range of their parent router ensuring random collisions. The total number of nodes (N_{node}^{TOTAL}) in WSN is show in parentheses in Table 4.3.

For each cluster-tree topology, we study effect of various number of data flows N_{flow} equal to 2 or 4, and the number of sources N_{source} of each data flow equal to 3 or 6. The other parameters of data flows such as $req\_period =$ 0.6 sec, $sample\_size = 120$ bits and $sample\_ack = 0$ are fixed. For each N_{flow} and N_{source} combination, a set of 20 instances is randomly generated and the scheduling algorithm is run for each of them. The mean number of tasks N_{task}, which represents the complexity of the problem, and mean time of the GLPK solver are shown in Table 4.3. The solving times, which exceed the time limit of 600 sec, are not encompassed in the mean time, and their number is shown in parentheses. The column $time_{compact}$ stands for the ILP formulation with objective function (4.7), which gives feasible and compact schedule while minimizing the sum of the start times. The column $time_{feasible}$ stands for the ILP formulation with objective function equal to 0 giving the feasible schedule in a shorter time. Note that in this case the problem does not require an optimal solution but only a first integer feasible solution.

4.6 Simulation study

IEEE 802.15.4 standard supports acknowledgement and retransmission mechanisms to minimize the influence of the communication errors coming from the unreliable and time-varying characteristics of wireless channels. The purpose of this section is to show how the maximum number of retransmissions (parameter $macMaxFrameRetries$ [9]) impacts the reliability of data transmission, the energy consumption of the nodes and the end-to-end communication delay, using the simulation study based on the IEEE 802.15.4/ZigBee Opnet simulation model (Section 3.3) that has been configured using the TDCS scheduling tool [50].

4.6.1 Simulation setup

The simulation scenario (illustrated in Figure 4.10) consists of 14 clusters and 23 TelosB motes forming a cluster-tree WSN. The TelosB [14] is a battery-powered wireless module widely used in WSNs. The following experiments

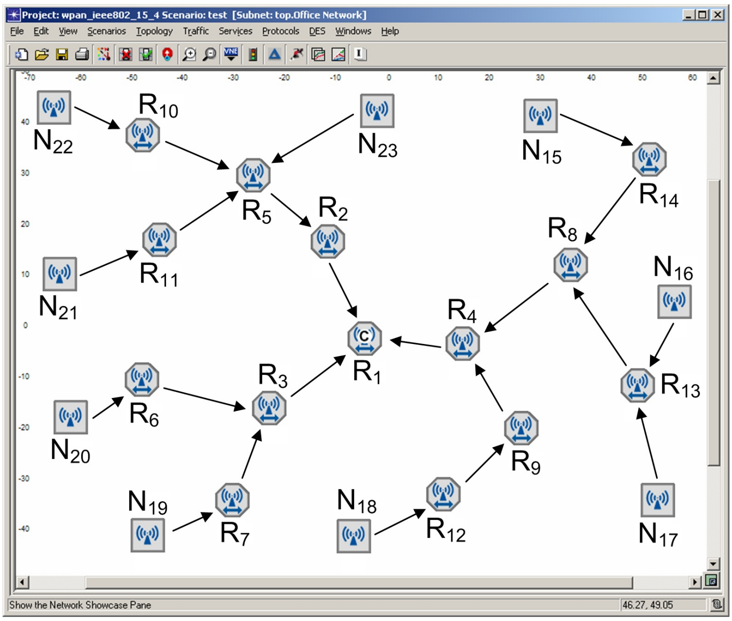

Project: wpan_ieee802_15_4 Scenario: test [Subnet: top.Office Network]
File Edit View Scenarios Topology Traffic Services Protocols DES Windows Help
N_{22}
R_{10}
N_{23}
N_{15}
R_{14}
R_5
R_2
R_8
R_{11}
N_{21}
N_{16}
R_4
c
R_1
R_3
R_{13}
R_6
N_{20}
R_9
N_{19}
N_{18}
R_7
R_{12}
N_{17}
Show the Network Showcase Pane
46.27, 49.05

Figure 4.10: The simulation scenario in Opnet Modeler (parent-child relationships).

considers the set of three time-bounded data flows of which user-defined parameters are summarized in Table 4.4.

New TDCS and configuration parameters of clusters, which ensure that each data flow meets its e2e deadlines while minimizing the energy consumption of the nodes, are generated for each number of retransmissions. Without loss of generality, the non-overlapping TDCSs are assumed (i.e. a single collision domain), because the simulation model does not support the definition of the multiple collision domains. The simulation time of one run is equal to 20 minutes involving generation of 1707 frames in case of flow 1, 1706 frames in case of flow 2 and 3585 frames in case of flow 3.

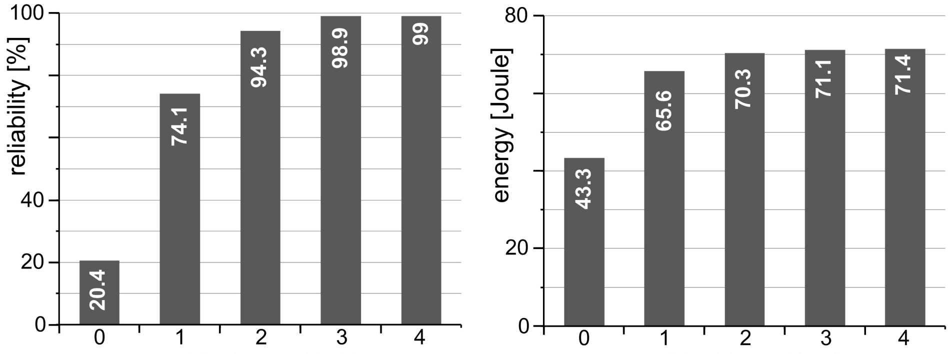
In fact, to engineer applications with certain guarantees, a certain confidence on the communication channel must be achieved, and this can be done by empirically analysing the channel error rate prior to a given deployment. For the sake of simplicity, the homogeneous channel error rate (a ratio of a number of dropped frames to a number of dispatched frames) equal to 20% is assumed. That means when a node receives a frame, the dropping probability is genereted as an uniformly distributed random number on the interval 0 to 100. If the dropping probability is less than 20, the received frame is dropped by a given node.

	flow 1	flow 2	flow 3
sources	$\{N_{19}, N_{21}, N_{23}\}$	$\{N_{17}, N_{18}\}$	$\{R_{12}, N_{16}, N_{20}\}$
sink	N_{15}	N_{20}	N_{22}
e2e_deadline [sec]	2.6	0.8	3.4
req_period [sec]	2.1	1.4	1
sample_size [bit]	64	32	48

Table 4.4: The user-defined parameters of the data flows from the simulation scenario.

4.6.2 Simulation results

Figure 4.11a shows the reliability of data transmission as a function of the maximum number of retransmissions (parameter *macMaxFrameRetries*). For each flow, the reliability of data transmission is calculated as the ratio of the number of dispatched frames by all sources to the number of received frames by the sink. The average ratio of all flows is then plotted in the chart.



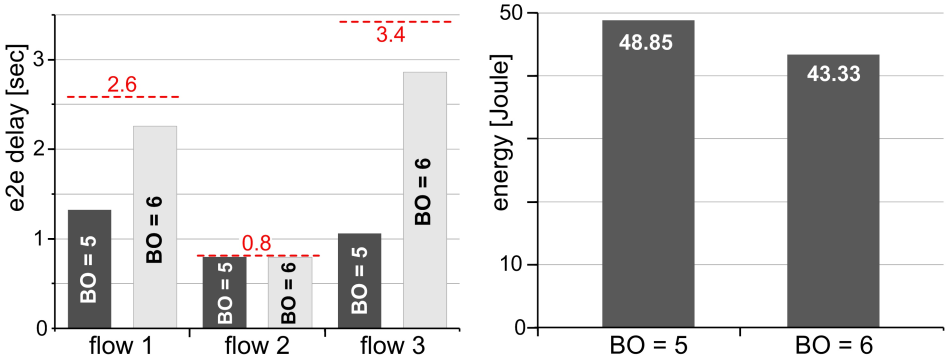
(a) Reliability (b) Energy consumption

Figure 4.11: Reliability of data transmission and sum of energy consumption of all nodes in the network.

Figure 4.11b shows the sum of energy consumption of all nodes within the simulation run as a function of the maximum number of retransmissions. As expected, the reliability and energy consumption grow with the maximum number of retransmissions. It can be observed that the reliability of acknowledged transmission with the maximum of one retransmission

($macMaxFrameRetries = 1$) increases 3.6 times against the reliability of unacknowledged transmission ($macMaxFrameRetries = 0$). On the other side, the energy consumption increases only 1.52 times.

In case of unacknowledged transmission, there exists two feasible TDCSs. A shorter TDCS with the period given by $BO = 5$, and a longer TDCS with the period given by $BO = 6$. Figure 4.12a confirms that both TDCSs are feasible, because the maximum end-to-end delays are shorter than end-to-end deadlines in both cases. However, Figure 4.12b shows that the network nodes consume more energy when the shorter TDCS ($BO = 5$) is applied. Hence, according to the required objectives, the TDCS scheduling tool returns the longer TDCS that meets all e2e deadlines while minimizing the energy consumption (i.e. maximizing the lifetime of the nodes).



(a) Maximum e2e delay (b) Energy consumption

Figure 4.12: Maximum e2e delay and sum of energy consumption of all nodes as a function of BO assuming unacknowledged transmission.

The maximum end-to-end delays (d_{ij}) for each flow and each number of retransmissions are presented in Figure 4.13. The dashed line at each column depicts the end-to-end deadline ($e2e\_deadline$) for a given flow. A first observation confirms that all TDCSs are feasible, because the maximum end-to-end delays are shorter than end-to-end deadlines. However, the e2e delays cannot be compare among each other, because new TDCS is generated from scratch for each number of retransmissions to meet required e2e deadlines. Note that for $macMaxFrameRetries = 5$ a feasible TDCS cannot be generated, because $BO_{min} = 7$ is greater than $BO_{max} = 6$. To increase BO_{max} to 7, the required period (parameter req_period) of all flows must be equal to or greater than 1.996608 sec, which is the value of BI for $BO = 7$ (Eq.(2.1)).

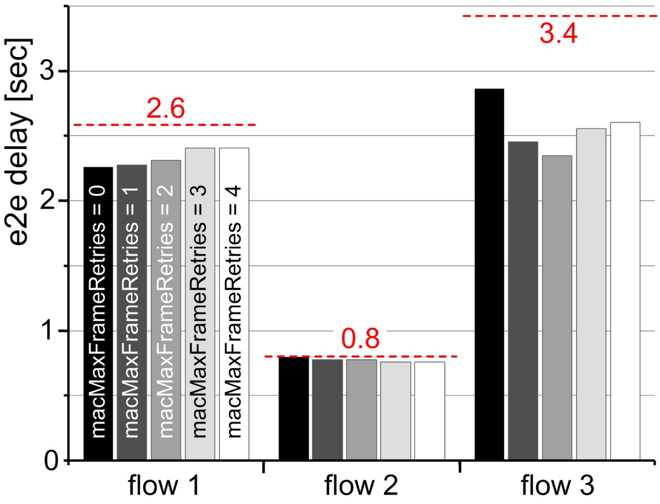


Figure 4.13: Maximum e2e delay as a function of the maximum number of retransmissions.

To obtain more illustrative results, the e2e deadline of flow 1 is reduced to 2.4 seconds and the other parameters are kept the same. In this case, a feasible TDCS can be only found for $macMaxFrameRetries$ in the range of 0 to 2, as depicted in Figure 4.14. For $macMaxFrameRetries = 3$ and up, no feasible TDCS exists, because the maximum e2e delay of a flow is always greater that its e2e deadline. Hence, it can be easily deduced that end-to-end delay grows with the maximum number of retransmissions as well.

4.7 Conclusions

The chapter shows how to minimize the energy consumption of the nodes by setting the beacon interval (TDCS period) as long as possible while respecting e2e deadlines of the flows and avoiding possible inter-cluster collisions. Binding of flows into one cluster-task is efficient with respect to the structure of superframe (dividing period BI into active portion and inactive portion) but also with respect to the complexity of the scheduling problem (volume of decisions is related to the square of potentially conflicting tasks). Note that grouping of GTSs in the transmit or receive direction leads to slightly pessimistic results (length of p_i^T or p_i^R is relatively short with respect to e2e deadline), but scheduling of separate GTSs would lead to dramatic increase of potentially conflicting tasks.

The solution is shown on iterative calls of the ILP algorithm, which

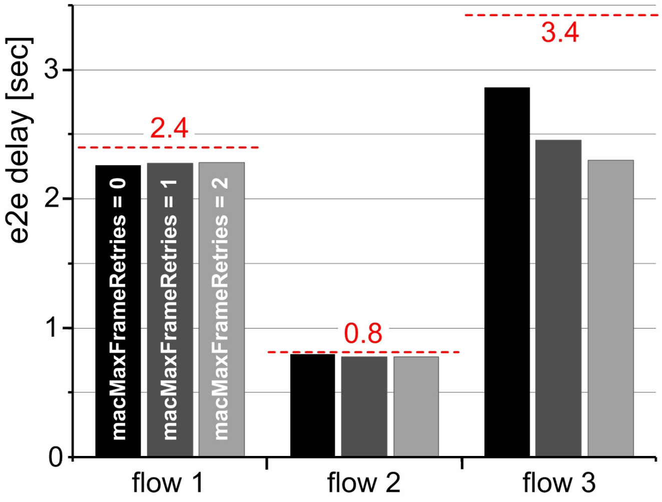


Figure 4.14: Maximum e2e delay as a function of the maximum number of retransmissions: reduced e2e deadline of flow 1.

gives precise mathematical formulation of the problem and shows acceptable performance for static configuration of middle-sized WSNs. The fact that a cluster is active only once during its period and the flows may have opposite directions leads to a cyclic formulation of the scheduling problem where one wave of a given data flow has to span over several periods. Thanks to the problem structure based on cyclic extension of RCPS/TC, it is quite straightforward to make cyclic extension of heuristic algorithms [52] that are able to handle RCPS/TC problems with up to one thousand of tasks in a few seconds.

An interesting issue to be investigated is the adaptive behaviour of the scheduling problem when new tasks are added to the original schedule. Such a problem should be solvable by fixing the start times of original tasks and using the same optimization algorithm. The time complexity in such a case should be related to the number of new tasks, thus allowing one to use optimal solvers.

The communication errors such as message corruption or message loss come from unreliable and time-varying characteristics of wireless channels. To increase the reliability of data transmission, the acknowledgement and retransmission mechanisms are employed. On the other side, the simulation results demonstrate that each retransmission also increases the energy consumption of the nodes and the end-to-end communication delay. Providing higher reliability while increasing the number of retransmissions requires greater amount of bandwidth that, consequently, enlarges the clusters active portions. On the other side, longer active portions imply higher duty-cycle and thus higher energy consumption of the nodes. In addition, longer clusters active portions may increase the TDCS period which induces longer end-to-end delays. Hence, the interdependence of reliability, energy consumption and timeliness make the network design more complex.

Using the proposed TDCS scheduling tool and simulation model, system designers are able to configure the IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs and easily find the trade-off between reliability, energy consumption and timeliness for a given application-specific implementation prior to the network deployment.

APPENDIX

4.A Table of symbols

The following table reports the symbols that are used through the Chapter 4, along with their definition.

Symbol	Definition
$A = (a_{ij})$	adjacency matrix describing parent-child tree
b	total number of nodes in the WSN
$C = (c_{ij})$	collision matrix
G	graph of communication tasks
N_{node}^{TOTAL}	total number of nodes in a WSN
N_{router}^{TOTAL}	total number of routers (clusters) in a WSN
n	number of tasks
p_i	processing time of task T_i
ptu	processing time unit
$\hat{q}_i$	segment of start time s_i
T_i	task i
s_i	start time of task T_i
$\hat{s}_i$	offset of start time s_i
$v_{i,j}$	offset weight
$w_{i,j}$	weight of the edge between tasks T_i and T_j
$\hat{x}_{ij}$	binary decision variable

Table 4.5: Table of symbols.

Chapter 5

Dimensioning and worst-case analysis of cluster-tree sensor networks

5.1 Introduction

The evaluation of the performance limits of Wireless Sensor Networks (WSNs) is of paramount importance to understand their behaviour under the worst-case conditions [53] and to make the appropriate design choices. This is particular relevant for time-sensitive WSN applications. Supporting time-sensitive WSN applications implies to predict and guarantee bounded end-to-end communication delays. To ensure bounded end-to-end delays and to avoid buffer overflow, network resources must be known in advance, and dimensioned along the path from a source to a sink. In that direction, this chapter aims at proposing a comprehensive methodology that provides a scientific approach for quick and efficient worst-case dimensioning of network resources (e.g. bandwidth and nodes' buffer size) to avoid their overflows and to minimize clusters' duty-cycle (reducing energy consumption of the nodes) in static or even dynamically changing WSNs with a cluster-tree topology, assuming bounded communication errors. Consequently, the worst-case performance bounds (e.g. the worst-case end-to-end delay) can be derived for a cluster-tree WSN with bounded resources. The analytical methodology is based on Network Calculus as a trade-off between complexity and accuracy. The main benefit of using Network Calculus is its generality and simplicity.

WSNs are commonly used for monitoring applications that gather sensory data in a central point called sink. In this chapter, the sink is considered as an autonomous entity that does not make part of the WSN, but can be associated to any of its routers through any (wired or wireless) communication means. Thus, the sink's mobility does not impact the WSN topology, but affects the destination of the data flows (any router in the WSN). However, while the statically associated sink is adequate for root centric WSN applications (e.g. an intruder alarm system delivering alerts to the control centre), other applications may impose or benefit from collecting data at different network locations (e.g. a doctor with a hand held computer collecting patients' status, a fire-fighter in a rescue mission or a mobile robot in a factory-floor).

Contribution

The main outcome of this chapter is the provision of a comprehensive methodology based on Network Calculus, which enables quick and efficient worst-case dimensioning of network resources (e.g. bandwidth and buffer size) in a static or even dynamically changing cluster-tree WSN where a static or mobile sink gathers data from all sensor nodes. Consequently, the worst-case performance bounds (e.g. end-to-end delay) can be evaluated for a cluster-tree WSN with bounded resources. The designed methodology presents two main components: (1) an analytical methodology for the worst-case dimensioning of network resources and delay bound analysis, (2) the impact of the sink's mobility on the worst-case performance of the cluster-tree network. This enables system designers to efficiently predict network resources that ensure a minimum QoS during extreme conditions (performance limits).

In particular, the Chapter 5 presents the following contributions:

1. A formulation of a simple yet efficient methodology, based on Network Calculus, to characterize incoming and outgoing data traffic in each router in the cluster-tree WSN (Section 5.5) and to derive upper bounds on buffer requirements and per-hop and end-to-end delays for both upstream and downstream directions (Section 5.6).
2. A description of how to instantiate this methodology in the design of IEEE 802.15.4/ZigBee cluster-tree WSNs, as an illustrative example that confirms the applicability of general approach for specific protocols (Section 5.7).
3. A demonstration of the validity of proposed methodology through an experimental test-bed based on Commercial-Off-The-Shelf (COTS)

technologies, where the experimental results are compared against the theoretical results and assess the pessimism of the theoretical model (Section 5.8).

4. An analysis of the impact of the sink mobility on the worst-case network performance and an outline of alternatives for sink mobility management, namely how routes must be updated upon the mobility of the sink, and how this procedure affects the worst-case network performance (network inaccessibility times) (Section 5.9).

Notations and symbols used in this chapter are summarized in Appendix 5.A.

5.2 Related work

The evaluation of the fundamental performance limits of WSNs has been addressed in several research works. In [53], the energy-constrained limits of WSNs with respect to the network throughput and operational lifetime has been evaluated. The authors have showed that with fixed node density, lifetime of WSN decreases in the order of $1/n$ as the number of deployed nodes n grows. Even with renewable energy sources, the maximum sustainable throughput in energy-constrained sensor networks scales worse than the capacity based on interference among concurrent transmissions, as long as the physical network size grows with n in an order greater than $\log n$. In [54], the authors have evaluated the real-time capacity of multi-hop WSNs, identifying how much real-time data the network can transfer by their deadlines. A capacity bound has been derived for load-balanced as well as load-unbalanced sensor networks using (ideal) MAC protocols with fixed priority packet scheduling mechanisms. The effects of various link layer multiplexing schemes such as time-division multiplexing and frequency-division multiplexing have been discussed. It has been shown that deadlines are never missed when the network capacity bound is not exceeded. Both above mentioned papers consider ad hoc WSNs. In [55], the authors have explored the fundamental limits for acceptable loads, utilization and delays in multi-hop sensor networks with fixed linear and grid topologies, in case all sensor nodes are equally capable to reach the sink (fair-access criterion). The upper bounds on network utilization and lower bounds on sensing time interval have been derived for any MAC protocol confirming to the fair-access criterion.

Another line of research works deals with soft real-time routing in WSNs. SPEED, MMSPEED and RPAR are some of the routing protocols providing soft end-to-end deadline guarantees in ad hoc WSNs. These protocols utilize location information to carry out routing decisions such that each node must be location-aware. SPEED [5] guarantees a uniform delivery speed all over the network so that the end-to-end delay of a message is proportional to the distance between source and sink. Thus, it is possible to predict if the end-to-end deadlines can be met or not. However, the SPEED protocol provides only one network-wide speed, which is not suitable for differentiating various traffic with different deadlines. MMSPEED [56] extends SPEED to support different delivery speeds and levels of reliability across the network, such that differentiated QoS can be achieved. Both SPEED and MMSPEED use fixed transmission power. Real-time Power-Aware Routing (RPAR) protocol [36] integrates transmission power control and real-time routing for supporting energy-efficient soft real-time communication. It is based on the assumption that a higher transmission power results in higher speed. The transmission power is increased if the required speed is not satisfied, otherwise if the required speed is satisfied the transmission power is decreased (to improve energy efficiency). On the contrary, this chapter assumes cluster-tree WSNs, where the routing decisions are simple and time-efficient because each node only interacts with its predefined parent/child nodes, and worst-case dimensioning is the main objective.

The worst-case analysis and resource dimensioning of WSNs using Network Calculus has been pursued by Schmitt et al., who proposed the Sensor Network Calculus methodology. In [57], Sensor Network Calculus was introduced and basic components such as arrival and service curves were defined. The system model assumes generic tree-based topologies with nodes transmitting sensor data towards the sink, that is associated to the root. The authors also proposed a general iterative procedure to compute the network internal flows and, subsequently, the resource requirements and the delay bounds. On the contrary, the recurrent equations are employed in this chapter so that to avoid iterative computations that are more complex and time consuming and not suitable for large scale WSNs. In [58], the previous Sensor Network Calculus framework was extended to incorporate in-network processing features (e.g. data aggregation). This chapter abstracts from the computational resources in the network nodes and from data aggregation. Lenzini et al. [59] have derived a tighter end-to-end delay bound for each single data flow in tree-based WSNs with FIFO multiplexing nodes. In [60], the authors searched for the worst-case topology (i.e. the topology that exhibits the worst-case behaviour in terms of buffer requirements, delay bounds and network lifetime) in networks with random nodes deployment. Finding the general worst-case topology is a complex task, thus their methodology explores the worst-case tree, constrained on maximum depth and number of child routers, that maximizes the arrival curve of the root node. As compared to the aforementioned papers, the system model presented in this chapter is more accurate for the specific case of cluster-tree topologies and the sink can be associated to any router in the WSN.

There have also been several research works on contention-free protocols and mechanisms based on resource reservations to achieve the desired QoS [61–63]. Caccamo et al. [61] have proposed a collision-free MAC protocol based on the decentralized earliest-deadline first (Implicit-EDF) packet scheduling algorithm. The key idea is to replicate the EDF schedule at each node to ensure contention-free packet transmission. If the schedules are kept identical, each node will know which one has the message with the shortest deadline and has the right to transmit next. However, it only works when the nodes are organized in hexagonal cells using frequencies different from any of their nearby cells, which requires transceivers supporting multiple frequencies. In addition, the nodes need tight clock synchronization and to know the characteristics of all periodic traffic a priori. These assumptions are uncommon in most WSN applications. Facchineti et al. [62] have presented a MAC protocol based on implicit-EDF to schedule real-time wireless communication in a network of mobile robots. They assumed the network is not fully linked and developed a consensus protocol to tolerate hidden nodes, allow dynamic schedule updates and dynamic node membership. Like implicit-EDF, their protocol relies on clock synchronization and cooperation among nodes. Each node needs to know the positions of all other nodes, and to take into account the number of hops a message needs to reach destination. The Robust Implicit-EDF (RI-EDF) protocol [63] also builds upon the EDF scheduling algorithm to derive a schedule for the network, implicitly avoiding a contention on the medium. Contrary to I-EDF, RI-EDF protocol assumes no clock synchronization among nodes and a fully linked network, and provides robustness in the presence of node failures or packet losses. This chapter assumes a TDMA-like policy for medium access, such as the Guaranteed Time Slot (GTS) in IEEE 802.15.4 protocol. Differentiation is made based on the flow specification by guaranteeing more time slots.

On the other hand, several research works addressed the use of sink mobility to minimize energy consumption in WSNs [64, 65]. The proposed approaches use random, predictable or controlled mobility of one or more sinks [64]. Four strategies (random, geographically, intelligent and genetic

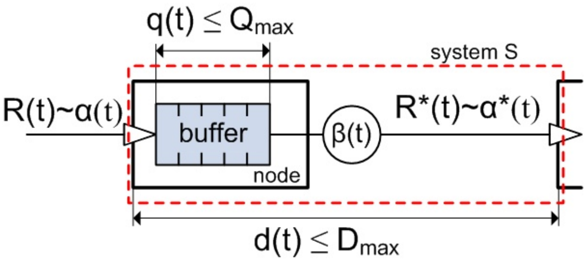


Figure 5.1: The basic system model in Network Calculus.

algorithm based) focusing on optimal sink placement for minimizing the worst-case delay as well as maximizing the lifetime of a WSN have been introduced in [65]. Conversely, the proposed methodology in this chapter computes the worst-case delays and resource requirements for any sink position.

5.3 Background on Network Calculus

Network Calculus [31] is a mathematical methodology based on min-plus algebra that applies to the deterministic analysis of queuing/flows in communication networks. This section briefly introduces the aspects of the Network Calculus formalism that are most significant to this paper. For additional details please refer to [31].

A basic system model S in Network Calculus consists of a buffered FIFO (First-In, First-Out order) node with the corresponding transmission link (Figure 5.1). For a given data flow, the input function $R(t)$ represents a cumulative number of bits that have arrived to system S in the time interval $(0, t)$. The output function $R^*(t)$ represents the number of bits that have left S in the same interval $(0, t)$. Both functions are wide sense increasing, i.e. $R(s) \le R(t)$ if and only if $s \le t$.

Guaranteeing performance bounds to a data flow requires that input function $R(t)$ and output function $R^*(t)$ be constrained. In Network Calculus these features are modelled by the concept of arrival and service curves.

Definition 1 Arrival Curve $\alpha(t)$ *(Figure 5.2). Let* $\alpha(t)$ *be a wide-sense increasing function for* $t \ge 0$*. Then an incoming data with input function* $R(t)$ *is upper bounded by* $\alpha(t)$ *iff for* $\forall s, 0 \le s \le t, R(t) - R(s) \le (t - s)$*. It is also said that* $R(t)$ *is* α *smooth or* $R(t)$ *is constrained by* $\alpha(t)$*, i.e.* $R(t) \sim \alpha(t)$*.*

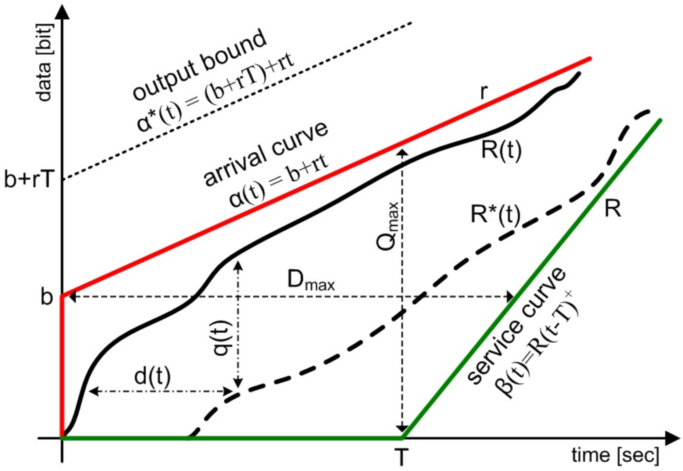


Figure 5.2: Example of a cumulative input function $R(t)$ constrained by affine arrival curve $\alpha_{b,r}(t)$ and a cumulative output function $R^*(t)$ constrained by rate-latency service curve $\beta_{R,T}(t)$.

Definition 2 Service Curve $\beta(t)$ *(Figure 5.2). Consider system S and a flow through S with input and output functions $R(t)$ and $R^*(t)$, respectively. Then S offers to the traversing flow a service curve $\beta(t)$ iff $\beta(t)$ is a wide-sense increasing function with $\beta(0) = 0$, and for $\forall t$ there exists $t_0 \leq t$ such that $R^*(t) - R^*(t_0) \geq \beta(t - t_0)$. This means that an outgoing data with output function $R^*(t)$ during any period $(t - t_0)$ is at least equal to $\beta(t - t_0)$.*

The knowledge of the arrival and service curves enables to determine the performance bounds for a lossless system, namely the delay bound D_{max}, which represents the worst-case delay of a message traversing the system S, and the backlog bound Q_{max}, which represents the worst-case queue length of a flow, i.e. indicates the minimum buffer size requirement inside the system S. Let a flow with input function $R(t)$, constrained by arrival curve $\alpha(t)$, traverses a system S that offers a service curve $\beta(t)$. It results that:

Definition 3 *The* Delay Bound D_{max} *is the maximum horizontal distance between $\alpha(t)$ and $\beta(t)$, and for $\forall t \geq 0$ the delay $d(t)$ satisfies:*

$$d(t) \leq \sup_{s \geq 0} \left\{ \inf \left\{ \tau \geq 0 : \alpha(s) \leq \beta(s + \tau) \right\} \right\} = D_{max} \tag{5.1}$$

Intuitively, D_{max} is the amount of time the arrival curve α must be shifted forward in time so that it lies below service curve β. From Eq. (5.1) it follows that $\beta_1 \leq \beta_2 \Rightarrow D_{max}(\beta_1) \geq D_{max}(\beta_2)$.

Definition 4 *The* Backlog Bound Q_{max} *is the maximum vertical distance between* $\alpha(t)$ *and* $\beta(t)$*, and for* $\forall t \geq 0$ *the backlog* $q(t)$ *satisfies:*

$$q(t) \leq \sup_{s \geq 0}\Big\{\alpha(s) - \beta(s)\Big\} = Q_{max} \tag{5.2}$$

In Network Calculus, it is also possible to express an upper bound for an outgoing data with output function $R^*(t)$, called *output bound.*

Definition 5 Output Bound $\alpha^*(t)$. *Assume that a flow with input function* $R(t)$*, constrained by arrival curve* $\alpha(t)$*, traverses a system* S *that offers a service curve* $\beta(t)$*. Then, the output function* $R^*(t)$ *is upper bounded by the following output bound* $\alpha^*(t)$*:*

$$\alpha^*(t) \leq (\alpha \odot \beta) \geq \alpha(t) \tag{5.3}$$

where $\odot$ *is the* min-plus deconvolution *defined for* $f, g \in \mathbb{F}$*, where* $\mathbb{F}$ *is the set of wide sense increasing functions, as:*

$$(f \odot g)(t) = \sup_{s \geq 0}\Big\{f(t+s) - g(s)\Big\} \qquad \text{for } \forall t \in \mathbb{R} \tag{5.4}$$

So far a system S has been handled as a single buffered node (Figure 5.1). However, system S might also be a sequence of nodes or even a complete network. In this case, the concatenation theorem enables to investigate a set of nodes in sequence as a single node.

Definition 6 Concatenation Theorem. *Assume a flow with input function* $R(t)$ *traversing system* S_1 *and* S_2 *in sequence, where* S_1 *offers service curve* $\beta_1(t)$ *and* S_2 *offers* $\beta_2(t)$*. Then the concatenation of these two systems offers the following single service curve* $\beta(t)$ *to the traversing flow:*

$$\beta(t) = (\beta_1 \otimes \beta_2)(t) = (\beta_2 \otimes \beta_1)(t) \tag{5.5}$$

where $\otimes$ *is the* min-plus convolution *defined for* $f, g \in \mathbb{F}$ *as:*

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t}\Big\{f(t-s) + g(s)\Big\} \qquad \text{for } \forall t \geq 0 \tag{5.6}$$

Min-plus convolution has several important properties, including being commutative and associative. Furthermore, convolution of concave curves is equal to their minimum [59]. It can be also shown that service curve β and arrival curve α can be expressed using a min-plus convolution as follows:

$$R^* \geq (R \otimes \beta)(t) = \inf_{0 \leq s \leq t} \Big\{ R(t-s) + \beta(s) \Big\} \qquad \text{for } \forall t \geq 0$$

$$R \leq (R \otimes \alpha)(t) = \inf_{0 \leq s \leq t} \Big\{ R(t-s) + \alpha(s) \Big\} \qquad \text{for } \forall t \geq 0$$

Due to the aggregation of the data flows in the direction of the sink, each router must provide a service curve $\beta(t)$ to the aggregated data flow. Thus, the delay and backlog bounds are computed for the aggregated data flow traversing the router. On the other hand, using the aggregate scheduling theorem, tighter bounds can be computed for each individual data flow traversing the network. In this work, both approaches (i.e. aggregated flow per router and individual flow over network) are used to compare the results.

Definition 7 Aggregate Scheduling Theorem. *Consider a lossless node multiplexing two data flows, 1 and 2, in FIFO order. Assume that flow 2 is constrained by the arrival curve $\alpha_2(t)$ and the node guarantees a service curve $\beta(t)$ to the aggregate of these two flows. Define the family of functions as:*

$$\beta_1^{eq}(t, \Theta) = \Big[\beta(t) - \alpha_2(t - \Theta) \Big]^+ \cdot 1_{\{t > \Theta\}} \tag{5.7}$$

where notation $1_{\{expr\}}$ is equal to 1 if expr *is true, and 0 otherwise, and $(x)^+$ denotes* $\max(0, x)$. *Then for $\forall \Theta \geq 0, \beta_1^{eq}(t, \Theta)$ is an equivalent service curve guaranteed for flow 1.*

So far an abstract Network Calculus model has been considered. The accuracy of the worst-case bounds depends on how tightly the selected arrival and service curves follow the real network behaviour. Different types of arrival and service curves have been proposed in Network Calculus (e.g. [31, 57]). However, the affine arrival curve and rate-latency service curve are the most used (as illustrated in Figure 5.2), since they lead to a fair trade-off between computing complexity and accuracy (approximation to the real system behaviour), as it will be shown in this chapter.

The *affine arrival curve* (Figure 5.2) is defined as $\alpha_{b,r}(t) = b + r \cdot t$ for $\forall t > 0$ and 0 otherwise, where b is called burst tolerance, which is the maximum number of bits that can arrive simultaneously at a given time to the system S, and r is the average data rate. This type of arrival curve represents a data traffic based on the average sensing rate with short-term fluctuations given by the burst tolerance, i.e. it allows a node to send b bits at once, but no more than an average of r bits per second over a long run.

The *rate-latency service curve* is defined as $\beta_{R,T}(t) = R \cdot (t - T)^+$, where $R \geq r$ is the guaranteed forwarding rate, T is the maximum latency of forwarding data (both depend on the nodes features, such as processing speed and resource allocation mechanism). If $r > R$, the bounds are infinite.

Hereafter, a system S is considered as the one that guarantees a rate-latency service curve $\beta_{R,T}(t)$ and that stores incoming data in a FIFO buffer. Then, the performance bounds D_{max} and Q_{max} (see Figure 5.2 for additional intuition) guaranteed to the data flow, constrained by the affine arrival curve $\alpha_{b,r}(t)$ and traversing system S, are easily computed as:

$$D_{max} = \frac{b}{R} + T \qquad\qquad Q_{max} = b + r \cdot T \tag{5.8}$$

Note that the first term b/R is interpreted as the part of the delay due to the burstiness of the incoming data, whereas T is due to the latency of the node.

An application of Eq. (5.3) to a data flow constrained by affine arrival curve $\alpha_{b,r}(t)$ and traversing system S guaranteeing a rate-latency service curve $\beta_{R,T}(t)$, the output bound of this data flow is expressed as (the proof can be found in [66]):

$$\alpha^*(t) = \alpha_{b,r}(t) \odot \beta_{R,T}(t) = \alpha_{b,r}(t) + r \cdot T = (b + r \cdot T) + r \cdot t = b^* + r^* \cdot t \tag{5.9}$$

According to the aggregate scheduling theorem, $\alpha_2(t)$ is a affine arrival curve and $\beta(t)$ is a rate-latency service curve, then an equivalent service curve for data flow 1 is expressed as:

$$\beta_1^{eq}(t, \Theta) = (R - r_2) \cdot \left[t - \Big(\frac{b_2 + r_2(T - \Theta)}{R - r_2} + T \Big) \right]^+ \cdot 1_{\{t > \Theta\}} \tag{5.10}$$

Hereafter, we omit repeating that arrive curves are affine and service curves are rate-latency, and that all curves are functions of time whenever doing so does not generate ambiguity.

5.4 System model

This section defines general cluster-tree topology and data flow models that will be considered in the following analysis. It also elaborates on the worst-case cluster scheduling; that is, the time sequence of clusters' active portions leading to the worst-case end-to-end delay for a message to be routed to the sink. To ensure predictable performance of a WSN, the network topology and data flows must be bounded. To provide closed-form recurrent expressions for computing the worst-case performance bounds in a WSN, the network topology and data load must be balanced.

5.4.1 Cluster-tree topology model

Like in Chapter 4, for achieving predictable resource guarantees, the cluster-tree is considered as a logical topology of WSNs. The cluster-tree topology model has been defined in Section 4.3.1. In this chapter, the routers and end-nodes are referred to as R_{ij} (i.e the j^{th} router at depth i) and N, respectively. The routers and end-nodes having sensing capabilities are generally referred to as *sensor nodes*. The depth of a node is defined as the number of logical hops from that node to the root. Note that the root is at depth zero and, by convention, trees grow down.

This section aims at specifying the *worst-case cluster-tree topology* which contains the maximum number of nodes in the network, i.e. the network topology configuration that leads to the worst-case performance. In the worst-case, when the maximum depth is reached, and all routers have the maximum number of associated child end-nodes and routers, the topology will be balanced (regular). However, a particular WSN can have unbalanced or even dynamically changing cluster-tree topology, but it can never exceed the worst-case topology, in terms of maximum depth and number of child routers/end-nodes. The irregularities in a particular topology introduce some pessimism to the analysis. On the other hand, for any network deployment can be found several cluster-tree topologies. Depending on the application, the system designer should select the most regular topology in design time to reduce the pessimism of the worst-case results.

The worst-case cluster-tree topology is defined by the following three parameters (derived from the ZigBee [10] specification):

$N^{MAX}_{end-node}$	Maximum number of end-nodes that can be associated to a router and have been allocated resource guarantees (e.g. time slots or bandwidth).

N_{router}^{MAX}	Maximum number of child routers that can be associated to a parent router and have been allocated resource guarantees.
H	Height of the tree, i.e. the maximum number of logical hops for a message from the deepest router to reach the root (including the root as a final hop). A network with only a root has a height of zero, and the maximum depth of an end-node is $H + 1$.

Note that a cluster-tree WSN may contain additional nodes per router than those defined by N_{router}^{MAX} and $N_{end-node}^{MAX}$ parameters. However, these additional nodes cannot be granted guaranteed resources.

Data gathering (all-to-one) and data dissemination (one-to-all) are two fundamental traffic patterns in WSNs [67]. This chapter only assumes the former, so called *convergecast*, where the sink gathers sensory data from all sensor nodes. The sink is considered as an autonomous and topology-independent static or mobile entity. The mobile behaviour means that a sink moves arbitrarily within a cluster-tree WSN and can be associated to any router within communication range. The router to which the sink is associated in a given moment is referred to as *sink router*. There can be more than one mobile sink in a WSN, but only one is active (i.e. gathers the sensory data) at a given time. Hence, another parameter, $H_{sink} \in (0, H)$, is specified to represent the maximum depth of the sink router in a cluster-tree topology, at a given moment. Note that the sink can be also statically attached to the root, i.e. $H_{sink} = 0$. In this case, the network contains only data links in upstream direction (i.e. from child nodes to the parent router) as was presented in [68].

The terminology and conventions are illustrated in Figure 5.3, corresponding to the worst-case configuration where $H = 2$, $N_{end-node}^{MAX} = 3$, $N_{router}^{MAX} = 2$, and $H_{sink} = 2$. The open arrows depict the data links in upstream or downstream direction.

5.4.2 Data flow model

This chapter assumes that all data traffic is routed to the sink router without any in-network processing on the way. In the worst-case, all sensor nodes are assumed to contribute equally to the network load, sensing and transmitting sensory data upper bounded by the affine arrival curve $\alpha_{data} = b_{data} + r_{data} \cdot t$ (Figure 5.4), where b_{data} is the burst tolerance and r_{data} is the average data rate. The affine arrival curve can represent any type of traffic, assuming that it can be bounded. It can represent a periodic or aperiodic traffic [69], or

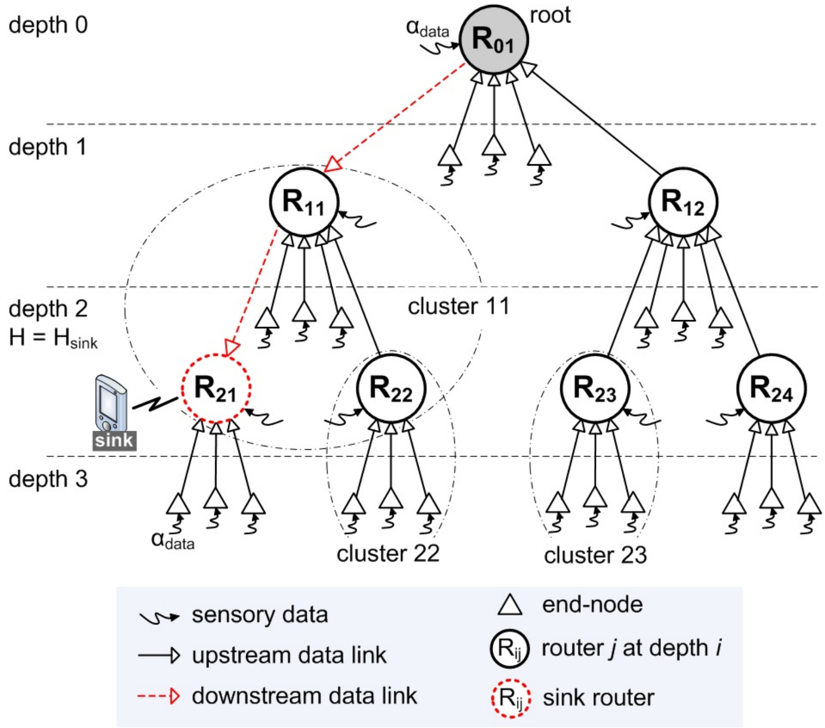


Figure 5.3: The worst-case cluster-tree topology model corresponding to a configuration where $N_{end-node}^{MAX} = 3$, $N_{router}^{MAX} = 2$, $H_{sink} = 2$, and $H = 2$.

any other random traffic (VBR traffic). This is the main reason for using this simple but effective and general arrival curve model: to be independent of any specific pattern/distribution of traffic.

In case of different sensory data traffic, α_{data} is considered to represent the upper bound of the highest sensory data traffic among all sensor nodes in the network. The analysis will lead to some pessimism if the variance between the highest sensory data traffic and the others is high, i.e. the pessimism increases with the variance. However, in many WSN applications the variance between the sensory data is likely to be small, since the sensing events are commonly reported by similar data types (e.g. single-precision floating-point number which occupies 32 bit).

Note that the data flows requiring real-time guarantees are only considered in the analysis. Other best-effort flows are also supposed to exist.

Each end-node is granted a service guarantee from its parent router corresponding to the rate-latency service curve $\beta_{end-node} = R_{end-node} \cdot (t - T_{end-node})^+$ (Figure 5.4), where $R_{end-node} \geq r_{data}$ is the guaranteed link

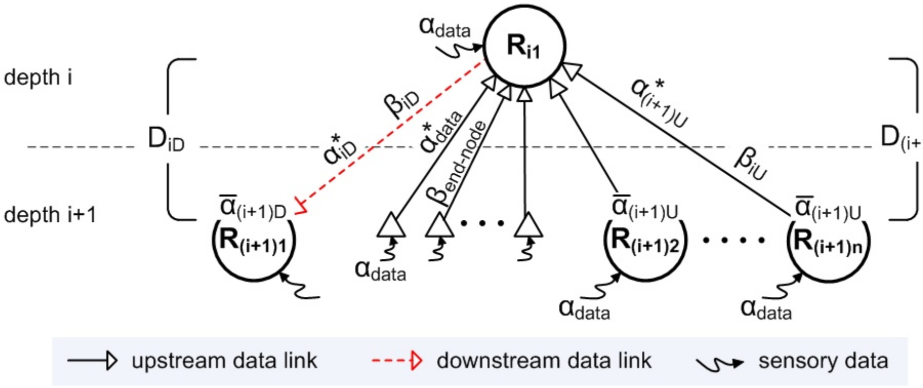


Figure 5.4: General data flow model with corresponding arrival curves, service curves and delays.

bandwidth and $T_{end-node}$ is the maximum latency of the service. The same service curve is provided to all end-nodes by their parent routers. Applying Eq. (5.9) to a flow constrained by the arrival curve $\alpha_{data}(t)$ and that is granted a service curve $\beta_{end-node}$ leads to the output bound α^*_{data}, which upper bounds the outgoing data from any end-node. It results that:

$$\alpha^*_{data} = \alpha_{data} + r_{data} \cdot T_{end-node} \tag{5.11}$$

On the other hand, the amount of bandwidth allocated by each router depends on the cumulative amount of data at its inputs, which increases towards the sink. Thus, the total input function $R(t)$ of each router depends on the depth, and consists of the sum of the output functions $R^*(t)$ of its end-nodes and child routers. Additionally, the router itself can be equipped with sensing capability producing a sensory data traffic bounded by α_{data}. Thus, in general case, the arrival curve constraining the total input function $R(t)$ of a router at a depth i is expressed as (Figure 5.4):

$$\bar{\alpha}_i = \alpha_{data} + N^{MAX}_{end-node} \cdot \alpha^*_{data} + N^{MAX}_{router} \cdot \alpha^*_{i+1} \tag{5.12}$$

This result can then be used in Eq. (5.9). The outgoing data of a router at depth i, that receives guaranteed service curve β_{i-1}, is constrained by the output bound as follows:

$$\alpha^*_i = \bar{\alpha}_i \oslash \beta_{i-1} \tag{5.13}$$

Hence, the data flow analysis consists in the computation of the arrival curves $\bar{\alpha}_i$ and output bounds α^*_i, using iteratively Eqs. (5.12) and (5.13), from the deepest routers until reaching the sink router. After that, the resource requirements of each router, in terms of buffer requirement Q_i and bandwidth requirement R_i, and the worst-case end-to-end delay bounds are computed.

In cluster-tree WSNs, where the sink can be associated to any router, data may flow in the upstream and downstream directions. In the upstream case, data is sent from the child nodes to its parent router (so called *upstream direction*), and the parent router must reserve enough bandwidth for the outgoing data of its child nodes. On the contrary, in the downstream case, data is sent from a parent router to its child router (so called *downstream direction*), and the parent router must reserve enough bandwidth for its own outgoing data. Note that if the sink is associated to the root, i.e. $H_{sink} = 0$, all data flows only in upstream direction. In what follows, the upstream and downstream directions are marked by the subscripts U and D, respectively (e.g. α^*_{iU}, $\bar{\alpha}_{iD}$). Each router at depth i provides two types of service curves (i.e. β_{iU} for data in the upstream direction and β_{iD} for data in the downstream direction) to its child routers at depth $i+1$, which are expressed as:

$$\beta_{iU} = R_{iU} \cdot (t - T_{iU})^+ \qquad \beta_{iD} = R_{iD} \cdot (t - T_{iD})^+ \tag{5.14}$$

where R_i is the guaranteed link bandwidth, and T_i is the maximum latency that a data must wait for a service. To ensure the balanced properties of the worst-case cluster-tree topology assumed in this methodology, the same upstream/downstream service curves must be guaranteed to all upstream/downstream data at a given depth. The *balanced properties* means that the worst-case cluster-topology and worst-case data flows are balanced such that the upstream/downstream routers in a given depth allocate and use the same resources. Note that the routers forwarding data in the upstream direction are referred to as *upstream routers* (e.g. R_{12} or R_{23} in the example in Figure 5.3) whereas the routers forwarding data in the downstream direction are referred to as *downstream routers* (e.g. R_{01} or R_{11}). In the same way, the data links between nodes are referred to as upstream or downstream.

5.4.3 Time Division Cluster Schedule

To avoid collisions between clusters, it is mandatory to schedule the clusters' active portions in an ordered sequence that is called *Time Division Cluster Schedule* (TDCS) (Section 4.4). In case of single collision domain, the TDCS must be non-overlapping, i.e. only one cluster can be active at any time. Hence, the period of TDCS is given by the number of clusters and the length of their active portions. On the contrary, in a network with multiple collision domains, the clusters from different non-overlapping collision domains may be active at the same time. It is easy to see that the period of overlapping TDCS is shorter (some active portions can run simultaneously) than the period of non-overlapping TDCS. Hence, the non-overlapping TDCS provides the worst-case performance bounds, which are the objective of this chapter. Note that the non-overlapping TDCS can be more pessimistic in networks with multiple collision domains.

The TDCS significantly affects the resource requirements and delay bounds in cluster-tree WSNs. The number of feasible non-overlapping TDCSs in a network with n routers is equal to the number of permutations given by n factorial ($n!$). Note that for each data flow originated in a given node, there is a corresponding best-case/worst-case TDCS that minimizes/maximizes the end-to-end delay of that flow, respectively. Thus, it is impossible to determine a general best-case or worst-case TDCS meeting the requirements of all data flows. On one hand, the best-case TDCS of a data flow originated in a node x comprises the consecutive sequence of active portions corresponding to the ordered sequence of the clusters traversed along the routing path from x to the sink. On the other hand, the worst-case TDCS (WC-TDCS) comprises the same ordered sequence of active portions, but in the reverse order, which means starting from the sink backward to the node x. The active portions of other clusters, which are not on the routing path, are appended before or after the previously formed sequence in arbitrary order such that a complete WC-TDCS for a given flow is produced. Using the proposed methodology based on the balanced properties of cluster-tree topology model (i.e. balanced topology with balanced load), the WSN resources are dimensioned for non-overlapping WC-TDCS of a data flow originated in the end-node that is farthest from the sink (i.e. a flow along the longest routing path in a WSN). Within a period of WC-TDCS the messages belonging to this data flow go only one hop forward. Note that for a particular cluster-tree WSN, the overlapping or non-overlapping application-specific TDCS, which achieves better performance bounds than WC-TDCS, can be found. However, the interest is in the worst-case performance bounds of general WSNs such that non-overlapping WC-TDCS is the objective.

Let us consider the example in Figure 5.3, where an end-node of router R_{24} sends sensory data to the sink (that is associated to router R_{21}), i.e. a data flow along the longest routing path in the WSN. The non-overlapping best-case TDCS for this flow comprises the continuous sequence of active portions of clusters 24, 12, 01 and 11. On the contrary, the WC-TDCS comprises the same sequence but in the reverse order, i.e. active portions of clusters 11, 01, 12 and 24. The active portions of other clusters (i.e. clusters 21, 22 and 23) are appended in arbitrary order such that the complete WC-TDCS is the sequence of active portions of clusters 22, **11**, **01**, **12**, **24**, 21 and 23, for example.

To reduce the resource requirements of the routers, the priority rule has been introduced in Section 4.4.1. Using this rule, the end-to-end delay of a data flow can be reduced by one period of TDCS at a router which handles the links in both directions for a given data flow.

5.5 Data flow analysis

This section serves as a basic building block for Section 5.6. Recurrent equations of the incoming or outgoing data in upstream or downstream direction as a function of the router's depth are derived, considering the general cluster-tree topology model presented in Section 5.4. It is assumed that the end-nodes have sensing capabilities, but the sensing capabilities of the routers are optional. Therefore, a binary variable ω is introduced. The value of ω is equal to 1 if routers have sensing capabilities; otherwise ω is equal to 0.

5.5.1 Upstream direction

First, the arrival curves of the incoming data in upstream direction $\bar{\alpha}_{iU}$ and the upper bounds of the outgoing data in upstream direction α^*_{iU} are evaluated depth by depth, using the Network Calculus methodology, starting from depth H (i.e. the deepest routers). These derivations are based on the work [68]. The analysis considers the general queuing model for the upstream direction as illustrated in Figure 5.5.

Analysis of depth H+1

At depth $H+1$, there is no router, there are only end-nodes generated sensory data constrained by the arrival curve α_{data}. A parent router at depth H guarantees the service curve $\beta_{end-node}$ to each of its end-nodes. Thus, according to Eq. (5.11) the outgoing data of each end-node is constrained by the output bound α^*_{data}.

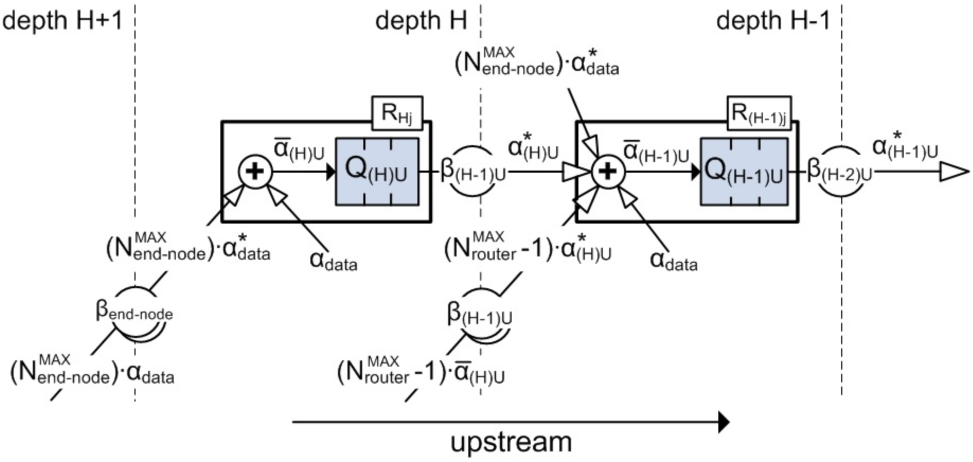

depth H+1
depth H
depth H-1
$(N^{MAX}_{end\text{-}node})\cdot\alpha^*_{data}$
R_{Hj}
$R_{(H-1)j}$
$\bar{\alpha}_{(H)U}$
$Q_{(H)U}$
$\beta_{(H-1)U}$
$\alpha^*_{(H)U}$
$\bar{\alpha}_{(H-1)U}$
$Q_{(H-1)U}$
$\beta_{(H-2)U}$
$\alpha^*_{(H-1)U}$
$(N^{MAX}_{end\text{-}node})\cdot\alpha^*_{data}$
α_{data}
$(N^{MAX}_{router}-1)\cdot\alpha^*_{(H)U}$
α_{data}
$\beta_{end\text{-}node}$
$\beta_{(H-1)U}$
$(N^{MAX}_{end\text{-}node})\cdot\alpha_{data}$
$(N^{MAX}_{router}-1)\cdot\bar{\alpha}_{(H)U}$
upstream

Figure 5.5: The queuing system model for the upstream direction.

Analysis of depth H

At depth H, the total incoming data of each router comprises the sum of the outgoing data of its $N^{MAX}_{end-node}$ end-nodes and, optionally, its own sensory data constrained by α_{data}. Thus, the arrival curve constraining the total incoming data is expressed as:

$$\bar{\alpha}_H = \omega \cdot \alpha_{data} + N^{MAX}_{end-node} \cdot \alpha^*_{data} \tag{5.15}$$

As a result, applying Eq. (5.11) will lead to:

$$\bar{\alpha}_H = \left(N^{MAX}_{end-node} + \omega\right) \cdot \alpha_{data} + N^{MAX}_{end-node} \cdot r_{data} \cdot T_{end-node} \tag{5.16}$$

where $\bar{r}_H = \left(N^{MAX}_{end-node} + \omega\right) \cdot r_{data}$ is the resulting aggregate arrival rate of the incoming data, and $\bar{b}_H = \left(N^{MAX}_{end-node} + \omega\right) \cdot b_{data} + N^{MAX}_{end-node} \cdot r_{data} \cdot T_{end-node}$ is the burst tolerance. Note that $\bar{\alpha}_H$ corresponds to the first two terms of the arrival curve (Eq. (5.12)) constraining the total incoming data $\bar{\alpha}_i$. These two terms are constant for upstream and downstream flows at each depth, hence $\bar{\alpha}_H$ is used without any directional subscripts (i.e. U or D).

The total incoming data upper bounded by $\bar{\alpha}_H$ is forwarded by a router at depth H to its parent router at depth $H-1$. This parent router guarantees a service curve $\beta_{(H-1)U} = R_{(H-1)U} \cdot \left(t - T_{(H-1)U}\right)^+$ to its child routers. Hence, according to Eq. (5.13), the output bound constraining the outgoing data from a router at depth H is then expressed as $\alpha^*_{(H)U} = \bar{\alpha}_H \oslash \beta_{(H-1)U}$. As a result, applying Eq. (5.9) will lead to:

$$\alpha^*_{(H)U} = \bar{\alpha}_H + \sigma_{(H-1)} \tag{5.17}$$

where $\sigma_{(H-1)} = \bar{r}_H \cdot T_{(H-1)U}$.

Analysis of depth H–1

The total incoming data of a router at depth $H-1$ comprises the outgoing data of its child router in addition to the data of its child end-nodes, and its own (optional) sensory data. Thus, the arrival curve constraining the total incoming data is expressed as $\bar{\alpha}_{(H-1)U} = \left(\omega \cdot \alpha_{data} + N^{MAX}_{end-node} \cdot \alpha^*_{data}\right) + N^{MAX}_{router} \cdot \alpha^*_{(H)U}$. As a result, using Eqs. (5.15) and (5.17) will lead to:

$$\bar{\alpha}_{(H-1)U} = \left(N^{MAX}_{router} + 1\right) \cdot \bar{\alpha}_H + N^{MAX}_{router} \cdot \sigma_{(H-1)} \tag{5.18}$$

The total incoming data upper bounded by $\bar{\alpha}_{(H-1)U}$ is forwarded by a router at depth $H-1$ to its parent routers at depth $H-2$. This parent router guarantees a service curve $\beta_{(H-2)U}$ to its child routers. Hence, according to Eq. (5.13), the output bound constraining the outgoing data from a router at depth $H-1$ is then expressed as $\alpha^*_{(H-1)U} = \bar{\alpha}_{(H-1)U} \odot \beta_{(H-2)U}$. As a result, applying Eqs. (5.3) and (5.18) will lead to:

$$\alpha^*_{(H-1)U} = \left(N^{MAX}_{router} + 1\right) \cdot \bar{\alpha}_H + N^{MAX}_{router} \cdot \sigma_{(H-1)} + \sigma_{(H-2)} \tag{5.19}$$

where $\sigma_{(H-2)} = \left(N^{MAX}_{router} + 1\right) \cdot \bar{r}_H \cdot T_{(H-2)U}$.

Analysis of depth H–2

Similarly to the previous case, the arrival curve constraining the total incoming data of a router at depth $H-2$ is expressed as:

$$\bar{\alpha}_{(H-2)U} = \begin{pmatrix} \left((N^{MAX}_{router})^2 + N^{MAX}_{router} + 1\right) \cdot \bar{\alpha}_H + \\ (N^{MAX}_{router})^2 \cdot \sigma_{(H-1)} + N^{MAX}_{router} \cdot \sigma_{(H-2)} \end{pmatrix} \tag{5.20}$$

The outgoing data forwarded from a router at depth $H-2$ to its parent router at depth $H-3$, providing a service curve $\beta_{(H-3)U}$, is constraining by the output bound:

$$\alpha^*_{(H-2)U} = \bar{\alpha}_{(H-2)U} + \sigma_{(H-3)} \tag{5.21}$$

where $\sigma_{(H-3)} = \big((N_{router}^{MAX})^2 + N_{router}^{MAX} + 1\big) \cdot \bar{r}_H \cdot T_{(H-3)U}$.

Analysis of general depth i

By recurrence, it can be easily proved that the arrival curve, constraining the total incoming data in upstream direction of each router at a given depth i, is expressed as follows:

$$\bar{\alpha}_{iU} = \sum_{j=0}^{H-i} (N_{router}^{MAX})^j \cdot \bar{\alpha}_H + \sum_{j=1}^{H-i} \big((N_{router}^{MAX})^j \cdot \sigma_{i+j-1}\big) \tag{5.22}$$

for $\forall i$, $0 \le i \le H$, where $\sigma_n = \sum_{k=0}^{H-(n+1)} (N_{router}^{MAX})^k \cdot \bar{r}_H \cdot T_{nU}$.

The output bound constraining the outgoing data in upstream direction from each child router at depth i, receiving a service curve $\beta_{(i-1)}$ from a parent router at depth $i-1$, is expressed as:

$$\alpha^*_{iU} = \bar{\alpha}_{iU} + \sigma_{i-1} = \sum_{j=0}^{H-i} (N_{router}^{MAX})^j \cdot \bar{\alpha}_H + \sum_{j=0}^{H-i} \big((N_{router}^{MAX})^j \cdot \sigma_{i+j-1}\big) \tag{5.23}$$

for $\forall i$, $0 < i \le H$.

5.5.2 Downstream direction

In this section, the arrival curves of the incoming data in downstream direction $\bar{\alpha}_{iD}$ and the upper bounds of the outgoing data in downstream direction α^*_{iD} is evaluated depth by depth, using the Network Calculus methodology, starting from depth 0 (i.e. the root). The analysis considers the general queuing model for downstream direction as illustrated in Figure 5.6.

Analysis of depth 0

At depth 0, there is only one router, the root, and its total incoming data comprises the sum of the outgoing data of its end-nodes, the sum of the outgoing data in upstream direction of its $(N_{router}^{MAX} - 1)$ child routers, and, optionally, its own sensory data constrained by α_{data}. Thus, the arrival curve

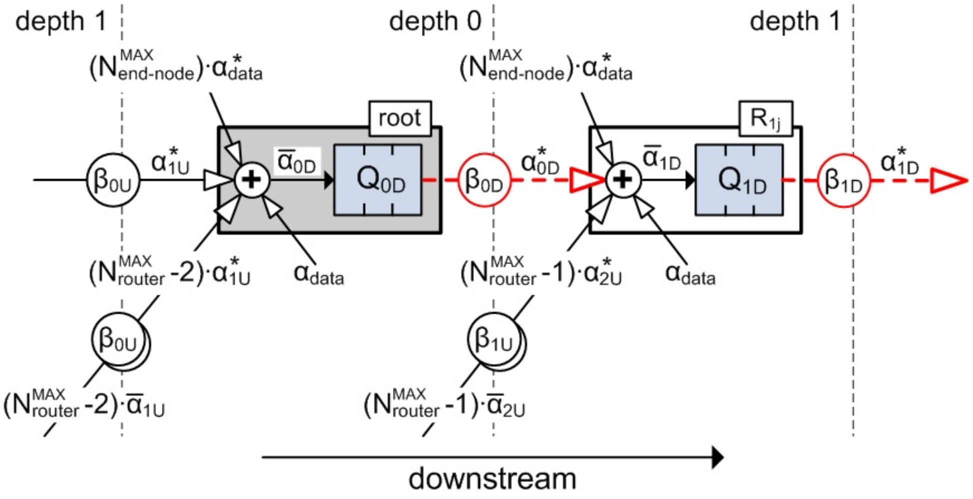


Figure 5.6: The queuing system model for downstream direction.

constraining the total incoming data is expressed as $\bar{\alpha}_{0D} = \bar{\alpha}_H + (N^{MAX}_{router} - 1) \cdot \alpha^*_{1U}$. As a result, applying Eq. (5.23) will lead to:

$$\bar{\alpha}_{0D} = (N^{MAX}_{router})^H \cdot \bar{\alpha}_H + (N^{MAX}_{router} - 1) \cdot \delta_0 \tag{5.24}$$

where $\delta_0 = \sum_{j=0}^{H-1} ((N^{MAX}_{router})^j \cdot \sigma_j)$.

The total incoming data upper bounded by $\bar{\alpha}_{0D}$ is forwarded by the root to one of its child routers in the sub-tree where the sink is associated. The root guarantees a service curve $\beta_{0D} = R_{0D} \cdot (t - T_{0D})^+$ to this child router at depth 1. According to Eq. (5.3), the outgoing data forwarded from the root at depth 0 to a child router at depth 1 is then constrained by the output bound:

$$\alpha^*_{0D} = \bar{\alpha}_{0D} \odot \beta_{0D} = \bar{\alpha}_{0D} + \tau_0 \tag{5.25}$$

where $\tau_0 = (N^{MAX}_{router})^H \cdot \bar{r}_H \cdot T_{0D}$.

Analysis of depth 1

The total incoming data of a router at depth 1 comprises the outgoing data of its parent router (i.e. the root, at depth 0) in addition to the outgoing data of its child end-nodes/routers, and, optionally, its own sensory data constrained by α_{data}. Thus, the arrival curve constraining the total incoming data is expressed as $\bar{\alpha}_{1D} = \bar{\alpha}_H + (N^{MAX}_{router} - 1) \cdot \alpha^*_{2U} + \alpha^*_{0D}$.

As a result, applying Eqs. (5.23) and (5.25) will lead to:

$$\bar{\alpha}_{1D} = \begin{pmatrix} ((N^{MAX}_{router})^{H} + (N^{MAX}_{router})^{H-1}) \cdot \bar{\alpha}_{H} + \\ \\ (N^{MAX}_{router} - 1) \cdot (\delta_0 + \delta_1) + \tau_0 \end{pmatrix} \tag{5.26}$$

where $\delta_1 = \sum_{j=0}^{H-2} ((N^{MAX}_{router})^{j} \cdot \sigma_{j+1})$.

The total incoming data upper bounded by $\bar{\alpha}_{1D}$ is forwarded by the router at depth 1 to one of its child routers in the sub-tree where the sink is associated. The parent router guarantees a service curve $\beta_{1D} = R_{1D} \cdot (t - T_{1D})^{+}$ to this child router at depth 2. According to Eq. (5.3), the outgoing data forwarded from the router at depth 1 to a child router at depth 2 is constrained by the output bound:

$$\alpha^{*}_{1D} = \bar{\alpha}_{1D} \odot \beta_{1D} = \bar{\alpha}_{1D} + \tau_1 \tag{5.27}$$

where $\tau_1 = ((N^{MAX}_{router})^{H} + (N^{MAX}_{router})^{H-1}) \cdot \bar{r}_{H} \cdot T_{1D}$.

Analysis of depth 2

Similar to the previous case, the arrival curve constraining the total incoming data of the router at depth 2 is expressed as $\bar{\alpha}_{2D} = \bar{\alpha}_{H} + (N^{MAX}_{router} - 1) \cdot \alpha^{*}_{3U} + \alpha^{*}_{1D}$. As a result, applying Eqs. (5.23) and (5.27) will lead to:

$$\bar{\alpha}_{2D} = \begin{pmatrix} ((N^{MAX}_{router})^{H} + (N^{MAX}_{router})^{H-1} + (N^{MAX}_{router})^{H-2}) \cdot \bar{\alpha}_{H} + \\ \\ (N^{MAX}_{router} -) \cdot (\delta_0 + \delta_1 + \delta_2) + \tau_0 + \tau_1 \end{pmatrix} \tag{5.28}$$

where $\delta_2 = \sum_{j=0}^{H-3} ((N^{MAX}_{router})^{j} \cdot \sigma_{j+2})$.

The outgoing data from the router at depth 2, guaranteeing a service curve $\beta_{2D} = R_{2D} \cdot (t - T_{2D})^{+}$, to a child router at depth 3 is upper bounded by the output bound:

$$\alpha^{*}_{2D} = \bar{\alpha}_{2D} \odot \beta_{2D} = \bar{\alpha}_{2D} + \tau_2 \tag{5.29}$$

where $\tau_2 = ((N^{MAX}_{router})^{H} + (N^{MAX}_{router})^{H-1} + (N^{MAX}_{router})^{H-2}) \cdot \bar{r}_{H} \cdot T_{2D}$.

Analysis of general depth i

By recurrence, the analysis is generalized for a general depth i. The arrival curve constraining the total incoming data in downstream direction of a router at a given depth i, for $i = 0,\ ,\ (H_{sink}\text{-}1)$, is then expressed as:

$$\bar{\alpha}_{iD} = \sum_{j=0}^{i} (N_{router}^{MAX})^{H-j} \cdot \bar{\alpha}_H + (N_{router}^{MAX} - 1) \cdot \sum_{j=0}^{i} \delta_j + \sum_{j=0}^{i-1} \tau_j \tag{5.30}$$

for $\forall i$, $0 \leq i \leq H$, where

$$\begin{aligned}
\delta_n &= \sum_{k=0}^{H-(n+1)} \left((N_{router}^{MAX})^k \cdot \sigma_{k+n}\right) \\
\sigma_n &= \sum_{k=0}^{H-(n+1)} (N_{router}^{MAX})^k \cdot \bar{r}_H \cdot T_{nU} \\
\tau_n &= \sum_{k=0}^{n} (N_{router}^{MAX})^{H-k} \cdot \bar{r}_H \cdot T_{nD}
\end{aligned}$$

The upper bound of the outgoing data in downstream direction from a parent router at depth i, guaranteeing a service curve β_{iD}, towards its child router at depth $i + 1$ is expressed as:

$$\begin{aligned}
\alpha_{iD}^* &= \bar{\alpha}_{iD} + \tau_i = \\
&\sum_{j=0}^{i} (N_{router}^{MAX})^{H-j} \cdot \bar{\alpha}_H + (N_{router}^{MAX} - 1) \cdot \sum_{j=0}^{i} \delta_j + \sum_{j=0}^{i} \tau_j
\end{aligned} \tag{5.31}$$

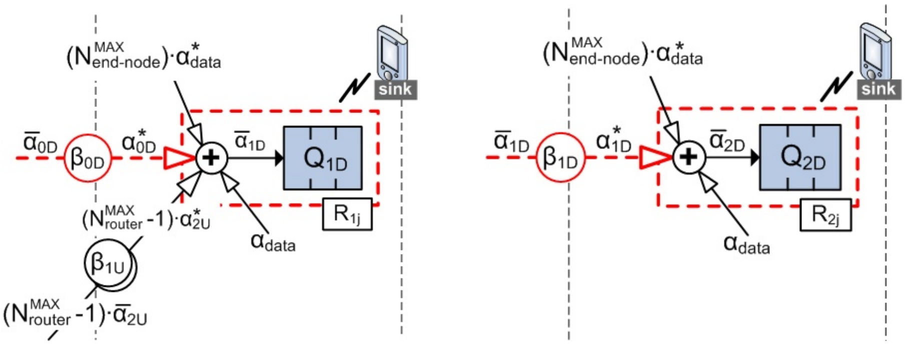
for $\forall i$, $0 \leq i < H_{sink}$.

Note that the sink can be associated to the router at a depth lower than the height of the cluster-tree, i.e. $H_{sink} < H$ (Figure 5.7a) or equal to the height of the cluster-tree, i.e. $H_{sink} = H$ (Figure 5.7b).

For $H_{sink} < H$, the arrival curve constraining the total incoming data is expressed as:

$$\bar{\alpha}_{(H_{sink})D} = \bar{\alpha}_H + N_{router}^{MAX} \cdot \alpha_{(H_{sink}+1)U}^* + \alpha_{(H_{sink}-1)D}^* \tag{5.32}$$

On the other hand, if $H_{sink} = H$, the arrival curve constraining the total



(a) $H_{sink} = 1 < H$ (depth=1) (b) $H_{sink} = 2 = H$ (depth=2)

Figure 5.7: The locations of a sink router and correspondent data flows.

incoming data is expressed as:

$$\bar{\alpha}_{(H_{sink})D} = \bar{\alpha}_H + \alpha^*_{(H_{sink}-1)D} \tag{5.33}$$

5.6 Worst-case network dimensioning

Supporting time-sensitive WSN applications implies to predict and guarantee bounded (worst-case) end-to-end communication delays. To ensure bounded end-to-end delays and to avoid buffer overflow, network resources must be known in advance, and dimensioned along the path from a source to a sink. In this section, the upper bounds on bandwidth and buffer requirements, and per-hop and end-to-end delays for both upstream and downstream directions are derived.

5.6.1 Per-router resources analysis

The aim is at specifying the minimum bandwidth of each upstream and downstream data links and the minimum buffer size at each router needed to store the bulk of data incoming through the router's inputs.

Bandwidth requirements

Consider a parent router at depth i providing a service curve β_{iU} or β_{iD} to its child routers at depth $i+1$ in upstream or downstream direction, respectively (see Figure 5.3).

In the upstream case, the outgoing data of a child router at depth $i+1$ is constrained by the output bound $\alpha^*_{(i+1)U}$ and dispatched through the upstream link to its parent router at depth i. Thus, to ensure a bounded delay, the guaranteed amount of bandwidth R_{iU} must be greater than or equal to the outgoing data rate $r^*_{(i+1)U}$. As a result, applying Eqs. (5.16) and (5.23) will lead to:

$$\begin{aligned} R_{iU} \geq r^*_{(i+1)U} = \bar{r}_{(i+1)U} = \sum_{j=0}^{H-(i+1)} (N^{MAX}_{router})^j \cdot \bar{r}_H = \\ \Omega_{iU}(H, N^{MAX}_{router}) \cdot (N^{MAX}_{end-node} + \omega) \cdot r_{data} \end{aligned} \tag{5.34}$$

for $\forall i$, $0 \leq i < H$.

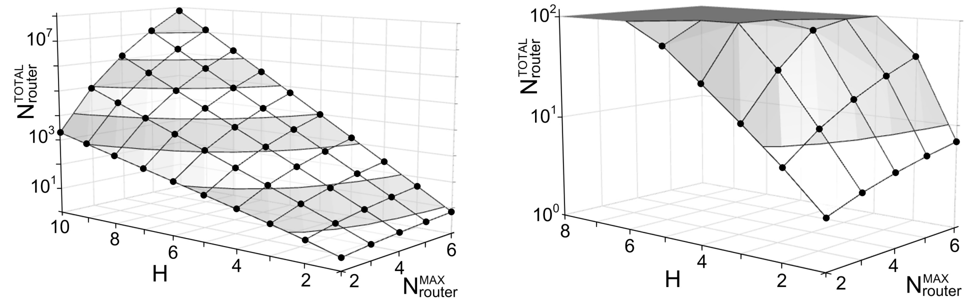
The $\Omega_{iU}(H, N^{MAX}_{router})$ is called the *upstream bandwidth increase factor* at a given depth i. This factor increases with the depth and N^{MAX}_{router}, and it represents the ratio of the additional bandwidth that a router, at a depth i, must guarantee to each of its child routers in the upstream direction as compared to the bandwidth required by a router at depth H.

In the downstream case, the total incoming data of the parent router at depth i is constrained by the arrival curve $\bar{\alpha}_{iD}$ and dispatched through a downstream link to its child router. Thus, to ensure a bounded delay, the guaranteed amount of bandwidth R_{iD} must be greater than or equal to the arrival rate of total input flow $\bar{r}_{iD}$. As a result, applying Eqs. (5.16) and (5.30) will lead to:

$$\begin{aligned} R_{iD} \geq \bar{r}_{iD} = r^*_{iD} = \sum_{j=0}^{i} (N^{MAX}_{router})^{H-j} \cdot \bar{r}_H = \\ \Omega_{iD}(H, N^{MAX}_{router}) \cdot (N^{MAX}_{end-node} + \omega) \cdot r_{data} \end{aligned} \tag{5.35}$$

for $\forall i$, $0 \leq i < H_{sink}$.

Similarly to the previous case, the $\Omega_{iD}(H, N^{MAX}_{router})$ is called the *downstream bandwidth increase factor* at a given depth i. This factor represents the ratio of the additional bandwidth that a router, at depth i, must guarantee to its child router in downstream direction as compared to the bandwidth required by a router at the depth H. Note that it is possible to determine the total number of routers in a cluster-tree WSN using the down-



(a) Total number of routers as a function of the height of the tree H and N_{router}^{MAX}.

(b) Feasible region for the total number of routers $N_{router}^{TOTAL} = 10^2$.

Figure 5.8: Total number of routers, i.e. downstream bandwidth increase factor at depth H (logarithmic scale).

stream bandwidth increase factor by having $i = H$, which is expressed as:

$$\Omega_{HD}(H, N_{router}^{MAX}) = N_{router}^{TOTAL} = \sum_{j=0}^{H} (N_{router}^{MAX})^{H-j} \tag{5.36}$$

Figure 5.8a presents the variation of the total number of routers N_{router}^{TOTAL} (i.e. the downstream bandwidth increase factor at depth H) as a function of the height of the tree H and the maximum number of child routers N_{router}^{MAX}.

It can be observed that if N_{router}^{MAX} is high (e.g. equal to 5) the impact of the height H on the total number of routers is very significant. Depending on the total number of routers allowed when dimensioning the WSN, high values of the N_{router}^{MAX} parameter can be tolerated if the maximum height of the tree is limited. For instance, if the cluster-tree WSN cannot tolerate more than 10^2 routers (see Figure 5.8b) all points in the X, Y, Z axis located below the plan defined by Z = 10^2 are potential solutions to determine the pair (H, N_{router}^{MAX}). For example, with this constraint, the height of the tree cannot exceed 2 if $N_{router}^{MAX} = 5$ or $N_{router}^{MAX} = 6$, while it can be set to 5 if $N_{router}^{MAX} = 2$.

Buffer requirements

At each router, the incoming data must be stored in a buffer before it is dispatched. To avoid buffer overflow, in the upstream case, the buffer of an upstream router at depth i must be able to store all incoming data, constrained by the arrival curve $\bar{\alpha}_{iU}$, until it is dispatched through the upstream link to the parent router at depth $i - 1$. The required buffer size Q_{iU} of an upstream router at depth i must be at least equal to the burst tolerance b^*_{iU} of the output bound α^*_{iU} (see Figure 5.2). Hence, applying Eq. (5.23) will lead to:

$$\begin{aligned} Q_{iU} = b^*_{iU} = b^{*\,burst}_{iU} + b^{*\,latency}_{iU} = \\ \sum_{j=0}^{H-i} (N^{MAX}_{router})^j \cdot \bar{b}_H + \sum_{j=0}^{H-i} \left((N^{MAX}_{router})^j \cdot \sigma_{i+j-1}\right) \end{aligned} \tag{5.37}$$

for $\forall i$, $0 \leq i \leq H$.

Observe that the upstream buffer requirement is the sum of two terms. The first term is the sum of burst tolerances of the sensory data b_{data} of all sensor nodes inside all sub-trees of a given router. The second term represents the cumulative effect of the service latency at each depth for upstream data.

In the downstream case, the buffer of a downstream router at depth i must be able to store all incoming data, constrained by the arrival curve $\bar{\alpha}_{iD}$, until it is dispatched through the downstream link to a child router at depth $i+1$. The required buffer size Q_{iD} of the downstream router at depth i must be at least equal to the burst tolerance b^*_{iD} of the output bound α^*_{iD} (see Figure 5.2). Hence, using Eq. (5.31) will lead to:

$$\begin{aligned} Q_{iD} = b^*_{iD} = b^{*\,burst}_{iD} + b^{*\,latency_U}_{iD} + b^{*\,latency_D}_{iD} = \\ \sum_{j=0}^{i} (N^{MAX}_{router})^{H-j} \cdot \bar{b}_H + (N^{MAX}_{router} - 1) \cdot \sum_{j=0}^{i} \delta_j + \sum_{j=0}^{i} \tau_j \end{aligned} \tag{5.38}$$

for $\forall i$, $0 \leq i < H_{sink}$.

Observe that the buffer requirement is the sum of three terms. Similarly to the upstream case, the first term is related to the burst tolerance b_{data}, and the second term is related to the cumulative effect of the service latencies of upstream data. The third term represents the cumulative effect of the service latency at each depth for downstream data. In case of a sink router at depth H_{sink}, the buffer requirement must be greater than or equal to the burst tolerance $\bar{b}_{(H_{sink})D}$ of total incoming data $\bar{\alpha}_{(H_{sink})D}$ given by Eq. (5.32) or Eq. (5.33).

5.6.2 End-to-end delay analysis

The worst-case end-to-end delay is the delay bound of a data flow router along the longest path in the network. It can be computed using two approaches, as follows.

Per-hop end-to-end delay

The first approach consists in computing the per-hop delay bounds of the aggregate flows, and then deducing the end-to-end delay bound as the sum of per-hop delays. In the upstream case, according to Eq. (5.8) the delay bound between a child router at depth i and its parent router at depth $i-1$ guaranteeing service curve $\beta_{(i-1)U}$ is expressed as $D_{iU} = \bar{b}_{iU}/R_{(i-1)U} + T_{(i-1)U}$. On the other hand, in the downstream case, the delay bound between a parent router at depth i, which guarantees service curve β_{iD} to its total incoming data constrained by arrival curve $\bar{\alpha}_{iD}$, and its child router at depth $i+1$ is expressed as $D_{iD} = \bar{b}_{iD}/R_{iD} + T_{iD}$. Hence, the worst-case end-to-end delay is the sum of all per-hop delay bounds along the longest routing path, as follows:

$$D_{e2e} = D_{end-node} + \sum_{i=1}^{H} D_{iU} + \sum_{i=0}^{H_{sink}-1} D_{iD} \tag{5.39}$$

where $D_{end-node} = b_{data}/R_{end-node} + T_{end-node}$ is the delay bound between an end-node and its parent router.

This approach is a bit pessimistic, since the delay bound at each router is computed for the aggregation of all incoming flows. Tighter end-to-end delay bounds can be computed for individual flows, as described next.

Per-flow end-to-end delay

The idea of this approach is to derive the service curves guaranteed to a particular individual flow f by the routers along the path, using the aggregate scheduling theorem in Eq. (5.10), and then deduce the network-wide service curve for flow f based on the concatenation theorem. Finally, according to Eq. (5.8), the end-to-end delay bound of a given flow f is computed using the network-wide service curve applied to the arrival curve of the incoming flow. The worst-case end-to-end delay is equal to the delay bound of a data flow along the longest routing path in the network. This technique has been used in [59].

Consider a consecutive sequence of routers along the longest routing path (e.g. from an end-node of router R_{24} to the sink router R_{21} in Figure 5.3). The per-flow approach to the worst-case end-to-end delay is based on the following algorithm.

1. Start from the sink router. If $H_{sink} = 0$ then the index variable $last = 0$ and the network-wide service curve $\beta_w = \beta_{lastU}$; else (i.e. $1 \leq H_{sink} < H$) $last = H_{sink} - 1$, $\beta_w = \beta_{lastD}$, and go to step 4.
2. The service curve β_w is guaranteed to the aggregate of incoming upstream flows of an upstream router at depth $last{+}1$ (i.e. N^{MAX}_{router} flows from child routers at depth $last + 2$, $N^{MAX}_{end-node}$ flows from end-nodes, and optional own sensory data). Using the aggregate scheduling in Eq. (5.10), the equivalent service curve β^{eq} is computed for the outgoing flow of a router at depth $last + 2$ upper bounded by $\alpha^*_{(last+2)U}$.
3. Using the concatenation theorem in Eq. (5.5), replace $\beta_w = \beta^{eq} \otimes \beta_{(last+1)U}$ since the concatenation is also service curve for the outgoing flow of a router at depth $last + 2$. The length of the router's tandem is then reduced by one. Increase the variable $last = last + 1$. If $last = H - 1$, then go to step 7; else go to step 2.
4. The service curve β_w is guaranteed to the aggregate of incoming upstream/downstream flows of the downstream router at depth $last$. Using the aggregate scheduling in Eq. (5.10), the equivalent service curve β^{eq} is computed for the incoming downstream flow of the downstream router upper bounded by $\alpha^*_{(last-1)D}$.
5. Using the concatenation theorem in Eq. (5.5), replace $\beta_w = \beta^{eq} \otimes \beta_{(last-1)D}$ since the concatenation is also service curve guaranteed to the outgoing downstream flow of the downstream router at depth $last - 1$. The length of the router's tandem is then reduced by one. Decrease the variable $last = last - 1$. If $last = 0$, then go to step 6; else go to step 4.
6. The service curve β_w is guaranteed to the aggregate of incoming upstream flows of the root. Using the aggregate scheduling in Eq. (5.10), the equivalent service curve β^{eq} is computed for the outgoing flow of an upstream router at depth $last + 1$ upper bounded by $\alpha^*_{(last+1)U}$. Then, using the concatenation theorem in Eq. (5.5), replace $\beta_w = \beta^{eq} \otimes \beta_{lastU}$ since the concatenation is also service curve for the outgoing upstream flow of an upstream router at depth $last + 1$. Go to step 2.
7. Using the aggregate scheduling in Eq. (5.10), the equivalent service curve β^{eq} is computed for the outgoing flow of an end-node at depth $H + 1$ upper bounded by α^*_{data}. Then, using the concatenation theorem in Eq. (5.5), the network-wide service curve $\beta_w = \beta^{eq} \otimes \beta_{end-node}$ guaranteed to the individual sensory data constrained by arrival curve α_{data} is computed.

8. Compute the end-to-end delay bound, Eq. (5.8), using the network-wide service β_w applied to the arrival curve α_{data} upper bounding a sensory data.

Section 5.8.3 experimentally proves that this latter approach provides tighter delay bounds than the former per-hop approach.

5.7 Application to IEEE 802.15.4/ZigBee

So far, the general methodology for providing timing and buffer guarantees in cluster-tree WSNs with mobile sink behaviour independently of any specific communication protocol has been analysed. This section shows how to apply the aforementioned general methodology to the specific case of IEEE 802.15.4/ZigBee cluster-tree WSNs. The IEEE 802.15.4/ZigBee [9,10] protocols stand as the leading communication technologies for WSNs and have been briefly introduced in Chapter 2.1.

The general cluster-tree topology (Section 5.4.1) can be represented by the particular ZigBee cluster-tree topology where ZigBee coordinator corresponds to the root, ZigBee router to the router and ZigBee end device to the end-node. The beacon-enabled mode is considered, since it supports cluster-tree topology and enables the provision of guaranteed bandwidth through the Guaranteed Time Slot (GTS) mechanism. Each GTS can be used to transfer time-sensitive data either in transmit direction, i.e. from child node to its parent router (upstream data link), or receive direction, i.e. from parent router to its child node (downstream data link). There can be allocated a maximum of 7 GTSs in each superframe. Hence, using this explicit GTS allocation, the maximum numbers of child routers and end-nodes (that require guaranteed bandwidth) associated to each router are constrained as follows $N_{router}^{MAX} + N_{end-node}^{MAX} \leq 7$. According to the IEEE 802.15.4 [9] standard, all clusters have the same duty-cycle. Hence, the WC-TDCS is given by the non-overlapping sequence of equally sized SDs, and the period of WC-TDCS is equal to BI.

5.7.1 Guaranteed bandwidth of a GTS

Each GTS includes effective data transmission and overheads (i.e. inter-frame spacing (IFS) and eventual acknowledgement and retransmissions - see Section 2.3). In practice, most WSN applications are likely to use frames that are smaller than the maximum allowed size of a MAC frame, which is equal to *aMaxPHYPacketSize* (1016 bits) [9]. Thus, in order to achieve more accurate results the parameter $MPDU_{max}$ representing the user-defined maximum size of MAC frames is introduced.

Next, the expression for the guaranteed bandwidth of one time slot, which is related to the maximum effective data transmission, is derived. The maximum time required for the whole transmission of a MAC frame (including the data frame, IFS and eventual acknowledgement and retransmission) is then expressed as:

$$T_{MPDU}^{MAX} = \begin{pmatrix} (macMaxFrameRetries \cdot \Omega + 1) \cdot \\ \left(\frac{MPDU_{max}}{C} + ack \cdot \Omega \right) + \Delta IFS \end{pmatrix} \tag{5.40}$$

where $MPDU_{max}$ is the user-defined maximum size of MAC frame, C is the data rate (assuming 250 kbps), ΔIFS is equal to SIFS or LIFS depending on the $MPDU_{max}$, ack stands for $macAckWaitDuration$, $\Omega = 1$ for an acknowledged transmission or $\Omega = 0$ for an unacknowledged transmission.

The maximum number of MAC frames with user-defined maximum size that can be transmitted during one time slot is expressed as:

$$N_{MPDU} = \left\lfloor \frac{TS}{T_{MPDU}^{MAX}} \right\rfloor \tag{5.41}$$

where TS is the duration of a time slot and is equal to $SD/16$. In the remaining time (i.e. $TS - N_{MPDU} \cdot T_{MPDU}$) , a frame smaller than $MPDU_{max}$ can only be transmitted if the whole transmission can be completed before the end of the GTS. The transmission time of the last frame is then expressed as:

$$T_{last\_MPDU} = \frac{TS - N_{MPDU} \cdot T_{MPDU}}{macMaxFrameRetries + 1} - \Delta IFS - ack \cdot \Omega \tag{5.42}$$

Finally, assuming a full duty-cycle (i.e. $SO = BO$) the guaranteed bandwidth of one GTS time slot is expressed as:

$$R_{TS}^{100\%} = \frac{N_{MPDU} \cdot MPDU_{max} + \max(T_{last\_MPDU}, 0) \cdot C}{\mathrm{SD}} \tag{5.43}$$

5.7.2 Characterization of the service curve

Each parent router must reserve a GTS with enough time slots for each of its child nodes. For upstream data links, the resulting bandwidth of GTS, guaranteed by a parent router at depth i and given by N_{iU}^{TS} time slots in transmit direction, must be greater than or equal to the total incoming arrival rate $\bar{r}_{(i+1)U}$ of a child node at depth $i+1$. On the contrary, for downstream data links, a parent router at depth i must reserve a GTS with N_{iD}^{TS} time slots in receive direction to its child router at depth $i+1$ such that the resulting link bandwidth is greater than or equal to its total incoming arrival rate $\bar{r}_{iD}$. It results that:

$$N_{iU}^{TS} = \left\lceil \frac{\bar{r}_{(i+1)U}}{R_{TS}} \right\rceil \qquad N_{iD}^{TS} = \left\lceil \frac{\bar{r}_{iD}}{R_{TS}} \right\rceil \qquad N_{end-node}^{TS} = \left\lceil \frac{\bar{r}_{data}}{R_{TS}} \right\rceil \tag{5.44}$$

Note that $N_{end-node}^{TS}$ is the number of GTS time slots guaranteed to each end-node by its parent router. Hence, a GTS with N_i^{TS} time slots provides rate-latency service $\beta_{R_i T_i}$, where $R_i = N_i^{TS} \cdot R_{TS}$ is the guaranteed bandwidth and T_i is the maximum latency that the data must wait to be served.

The service latencies T_i depend on the TDCS such that their worst-case values are achieved for the non-overlapping WC-TDCS (i.e. a TDCS of a data flow along the longest routing path in a WSN). Since the proposed methodology is based on the balanced properties of the cluster-tree topology model, the same service latency, equal to the worst-case one at a given depth, is provided to all data links at a given depth in upstream/downstream direction. Let us consider the example in Figure 5.3, where an end-node of router R_{24} sends sensory data to the sink associated to the router R_{21} (i.e. a flow along the longest routing path). The corresponding WC-TDCS may be given by the following sequence of active potions of clusters 11, 01, 12, 24, 23, 21 and 22 (Figure 5.9), for example. The worst-case service latencies at each depth, except depth 0, are given by the distance between the active portions of consecutive clusters on the longest routing path to the sink. At depth 0, the priority rule (Section 4.4.1) is applied. The period of WC-TDCS is equal to the time which spans between two consecutive active portions of the same cluster (i.e. BI).

Note that the service latencies of any application-specific or overlapping TDCS will be equal or shorter than the latencies of non-overlapping WC-TDCS. In these cases, the worst-case latency at a given depth is equal to the longest latency in upstream/downstream direction at this depth, which does not to be equal to the one along the longest routing path in a WSN.

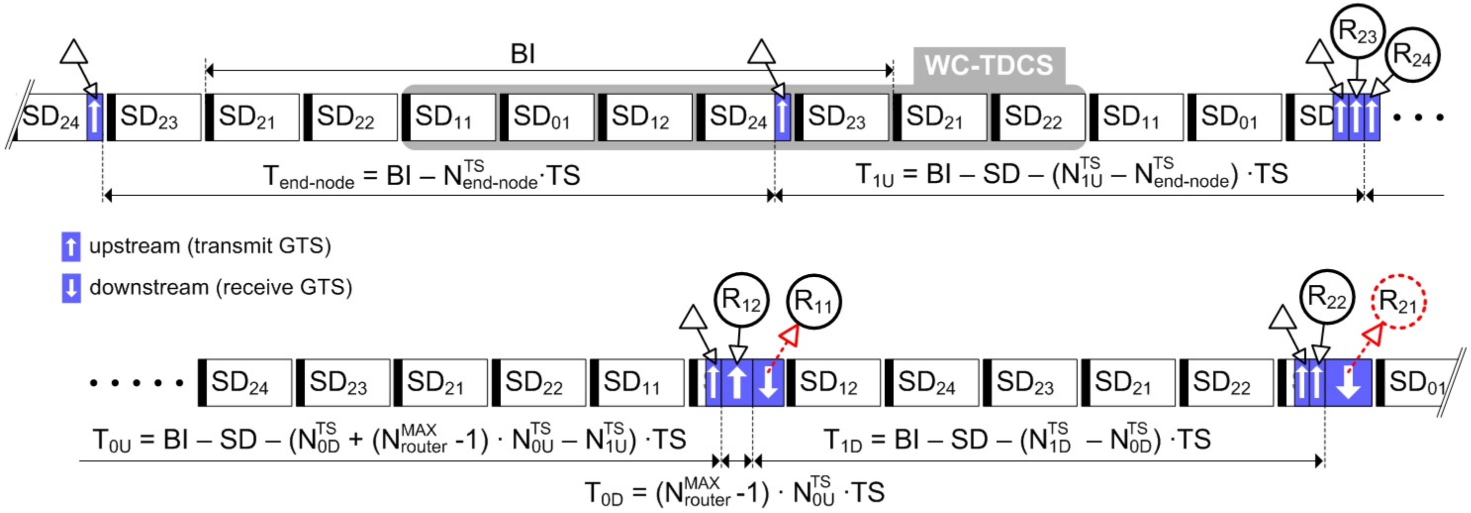

BI
WC-TDCS
R_{23}
R_{24}
SD_{24} SD_{23} SD_{21} SD_{22} SD_{11} SD_{01} SD_{12} SD_{24} SD_{23} SD_{21} SD_{22} SD_{11} SD_{01} SD
$T_{end\text{-}node} = BI - N^{TS}_{end\text{-}node} \cdot TS$
$T_{1U} = BI - SD - (N^{TS}_{1U} - N^{TS}_{end\text{-}node}) \cdot TS$
upstream (transmit GTS)
downstream (receive GTS)
R_{12}
R_{11}
R_{22}
R_{21}
SD_{24} SD_{23} SD_{21} SD_{22} SD_{11} SD_{12} SD_{24} SD_{23} SD_{21} SD_{22} SD_{01}
$T_{0U} = BI - SD - (N^{TS}_{0D} + (N^{MAX}_{router} - 1) \cdot N^{TS}_{0U} - N^{TS}_{1U}) \cdot TS$
$T_{1D} = BI - SD - (N^{TS}_{1D} - N^{TS}_{0D}) \cdot TS$
$T_{0D} = (N^{MAX}_{router} - 1) \cdot N^{TS}_{0U} \cdot TS$

Figure 5.9: The worst-case service latencies of a flow along the longest routing path related to the example in Figure 5.3.

According to Figure 5.9, the worst-case service latency guaranteed to a flow over an upstream data link at a given depth is expressed as:

- the latency guaranteed by a router to its end-node:

$$T_{end-node} = \mathrm{BI} - N^{TS}_{end-node} \cdot TS$$

- the latency guaranteed by a router at depth i to a child router at depth $i+1$, for $\forall i$, $0 < i < H$:

$$T_{iU} = \mathrm{BI} - \mathrm{SD} - \left(N^{TS}_{iU} - N^{TS}_{(i+1)U}\right) \cdot TS$$

- the latency guaranteed by the router at depth 0 to the child router at depth 1:

$$T_{0U} = \mathrm{BI} - \mathrm{SD} - \Big(N^{TS}_{0D} + (N^{MAX}_{router} - 1) \cdot N^{TS}_{0U} - N^{TS}_{1U}\Big) \cdot TS$$

On the other hand, the worst-case service latency guaranteed to a flow over a downstream data link at a given depth is expressed as:

- the latency guaranteed by a router at depth 0 to the child router at depth 1 (priority rule, Section 5.4.3):

$$T_{0D} = (N^{MAX}_{router} - 1) \cdot N^{TS}_{0U} \cdot TS$$

- the latency guaranteed by a router at depth i to the child router at depth $i+1$, for $\forall i$, $0 < i < H_{sink}$:

$$T_{iD} = \mathrm{BI} - \mathrm{SD} - \left(N^{TS}_{iD} - N^{TS}_{(i-1)D}\right) \cdot TS$$

5.7.3 IEEE 802.15.4/ZigBee cluster-tree WSN setup

The experimental scenario considers a simple cluster-tree WSN corresponding to the configuration where $H = 2$, $N^{MAX}_{end-node} = 1$, $N^{MAX}_{router} = 2$. For the sake of simplicity, only end-nodes are equipped with sensing capability (i.e. $\omega = 0$) and generate sensory data bounded by the arrival curve α_{data}. The SO is assumed to be equal to 4, which is the minimum value that is possible to use without resulting into synchronization problem [70], using open-ZB protocol stack [71] over TinyOS [72] and MICAz/TelosB motes. This constraint results from the lack of task prioritization and non-preemptive behaviour of the TinyOS operating system. According to Eq. (5.36), the total number of routers is equal to 7. Hence, BO must be set such that at least seven SDs with $SO = 4$ can fit inside the BI without overlapping.

In general, it results that:

$$BI \geq \Omega_{HD}(H, N_{router}^{MAX}) \cdot SD \Leftrightarrow \\ BO_{min} = \left\lceil \log_2(\Omega_{HD}(H, N_{router}^{MAX}) \cdot 2^{SO}) \right\rceil \quad (5.45)$$

As a result for $SO = 4$, the minimum BO is equal to 7, such that a maximum of $2^7/2^4 = 8$ SDs can fit in one BI. The maximum duty-cycle of each cluster is then equal to $2^{SO}/2^{BO} = 1/8 = 12.5\%$. Note that to maximize the lifetime of a WSN, the lowest duty-cycles must be chosen (IEEE 802.15.4 supports duty-cycles under 1%). As a result, the inactive portion is extended, and the nodes may stay in low power mode longer to save energy resources. On the other hand, low duty-cycles enlarge end-to-end delays. Hence, long lifetime is in contrast to the fast timing response of a WSN, so a trade-off must be found. In the example with $SO = 4$, the follwing duty-cycles can be achieved: 12.5% ($BO = 7$), 6.25% ($BO = 8$), 3.125% ($BO = 9$), and so on.

According to [9], the minimum CAP (i.e. *aMinCAPLength* parameter), which ensures that commands and best-effort data can still be transferred when GTSs are being used, is equal to 7.04 ms, assuming the 2.4 GHz ISM band, which corresponds to 1 time slot with $SO = 4$. Note that the CAP requires minimum 8, 4, 2 or 1 time slots with $SO = 0, 1, 2$ or 3, respectively. The remaining slots can be allocated for GTSs. Hence, the maximum CFP length is equal to $L_{CFP} = 15$ time slots. With this constraint, a router cannot reserve more than L_{CFP} time slots for 7 GTSs maximum, i.e. for its $N_{end-node}^{MAX}$ end-nodes and N_{router}^{MAX} child routers. Assuming that each end-node requires allocation of a GTS with $N_{end-node}^{TS}$ time slots (i.e. $r_{data} \leq N_{end-node}^{TS} \cdot R_{TS}$) from its parent router. Then, each child router can allocate a GTS with the maximum number of time slots equal to $\left\lfloor (L_{CFP} - N_{end-node}^{TS} \cdot N_{end-node}^{MAX})/N_{router}^{MAX} \right\rfloor$. According to Eqs. (5.34) and (5.35), the arrival rate r_{data} must be limited in order not to exceed the maximum bandwidth that a parent router can reserve. Obviously, due to the cumulative flow effect, the maximum bandwidth will be required either by the child routers of the root, in case the sink is associated to the root (i.e. $H_{sink} = 0$), or by the sink router, in other cases (i.e. $1 \leq H_{sink} \leq H$).

Thus, for $H_{sink} = 0$, the bandwidth guaranteed by the root to its child routers at depth 1 is expressed as:

$$R_0 = \left\lfloor \frac{L_{CFP} - N_{end-node}^{TS} \cdot N_{end-node}^{MAX}}{N_{router}^{MAX}} \right\rfloor \cdot R_{TS}$$

As a result, applying Eq. (5.34) will lead to the maximum arrival rate of the sensory data:

$$r_{data}^{MAX} = \left\lfloor \frac{L_{CFP} - N_{end-node}^{TS} \cdot N_{end-node}^{MAX}}{N_{router}^{MAX}} \right\rfloor \cdot \frac{R_{TS}}{\left(\sum_{j=0}^{H-1} (N_{router}^{MAX})^{j}\right) \cdot \left(N_{end-node}^{MAX} + \omega\right)} \qquad (5.46)$$

for $H_{sink} = 0$.

On the other hand, for $1 \leq H_{sink} \leq H$, the corresponding link bandwidth guaranteed by the parent router at depth $(H_{sink} - 1)$ to the sink router at depth H_{sink} is equal to:

$$R_{(H_{sink}-1)} = \left\lfloor \frac{L_{CFP} - N_{end-node}^{TS} \cdot N_{end-node}^{MAX}}{N_{router}^{MAX}} \right\rfloor \cdot R_{TS}$$

As a result, applying Eq. (5.35) will lead to the maximum arrival rate of the sensory data:

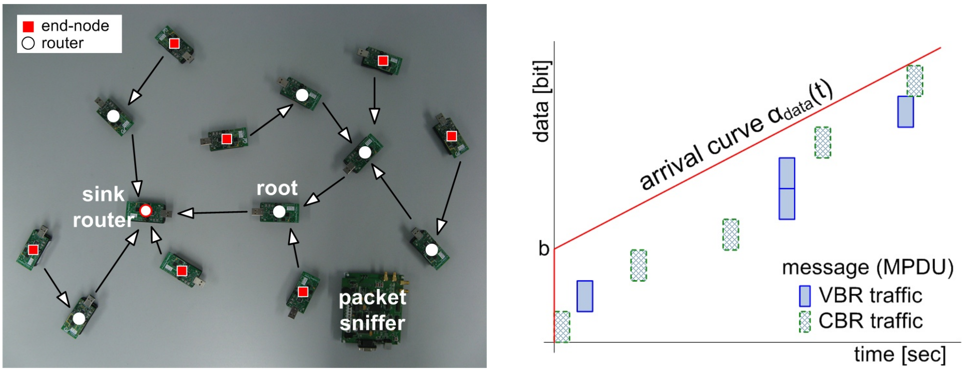
$$r_{data}^{MAX} = \left\lfloor \frac{L_{CFP} - N_{end-node}^{TS} \cdot N_{end-node}^{MAX}}{N_{router}^{MAX}} \right\rfloor \cdot \frac{R_{TS}}{\left(\sum_{j=0}^{H_{sink}-1} (N_{router}^{MAX})^{H-j}\right) \cdot \left(N_{end-node}^{MAX} + \omega\right)} \qquad (5.47)$$

for $\forall H_{sink}$, $1 \leq H_{sink} \leq H$.

The average arrival rate r_{data} of sensory data must be lower than r_{data}^{MAX} in any case. The value of burst b_{data} is selected according to the burstiness of sensory data.

5.8 Performance evaluation

This section compares the analytical results based on Network Calculus with the experimental results obtained through the use of IEEE 802.15.4/ZigBee technologies. The analytical results are computed using a Matlab tool [73], and the experimental results are obtained using a test-bed based on the TelosB motes [14].



(a) The test-bed deployment for $H_{sink} = 1$.

(b) The sensory data traffic.

Figure 5.10: The test-bed deployment and sensory data traffic upper bounded by arrival curve α_{data}.

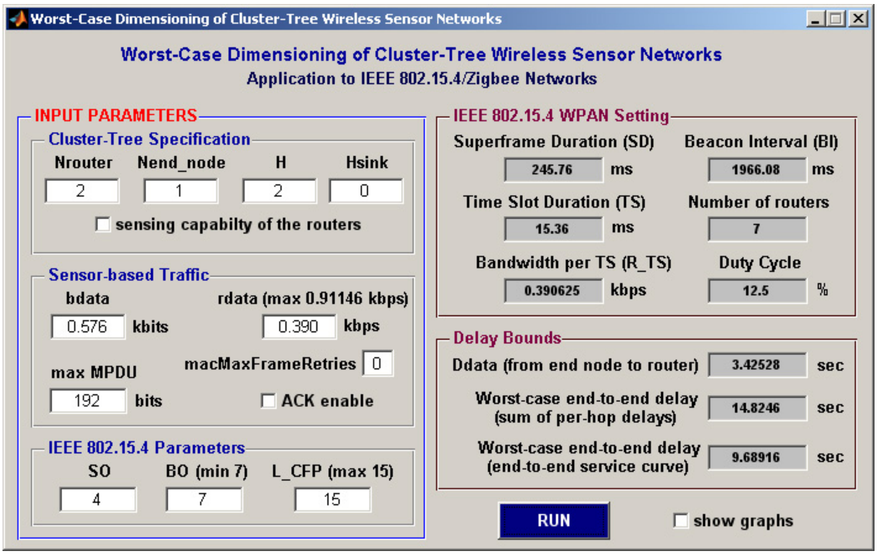
5.8.1 Network setup

The experimental test-bed (illustrated in Figure 5.10a) consists of 7 clusters and 14 TelosB motes running the TinyOS 1.x [72] operating system with open source implementation of the IEEE 802.15.4/ZigBee protocol stack [71]. The TelosB is a battery-powered wireless module with integrated sensors, IEEE 802.15.4 compliant radio, antenna, low power 16-bit RISC microcontroller, and programming capability via USB. For debugging purposes, it has been used the Chipcon CC2420 packet sniffer [74] that provides a raw list of the transmitted packets, and the Daintree Sensor Network Analyzer (SNA) [75] that provides additional functionalities, such as displaying the graphical topology of the network.

Note that, in practice, this experimental deployment could span over a wider region than the one illustrated in Figure 5.10a, provided that every end-node and child router is within radio range of its parent router (TelosB radio range is around several tens meters). Number of end-nodes associated to each router can also be higher (not all nodes might need guaranteed bandwidth).

The analytical model [73] was developed in Matlab, and can run in Command Line Interface (CLI) mode or Graphical User Interface (GUI) mode. On the left hand side of the GUI in Figure 5.11, the network setting and parameters of sensory data are entered. After the computation, the results and, optionally, several charts are shown on the right hand side. The values in Figure 5.11 correspond to the under mentioned network setting and the results from Section 5.8.3, namely the worst-case end-to-end delays for $H_{sink} = 0$.

The application running on the sensor nodes are configured to generate


Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks
RUN

Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks
Application to IEEE 802.15.4/Zigbee Networks

INPUT PARAMETERS

Cluster-Tree Specification
Nrouter: 2
Nend_node: 1
H: 2
Hsink: 0
☐ sensing capabilty of the routers

Sensor-based Traffic
bdata: 0.576 kbits
rdata (max 0.91146 kbps): 0.390 kbps
max MPDU: 192 bits
macMaxFrameRetries: 0
☐ ACK enable

IEEE 802.15.4 Parameters
SO: 4
BO (min 7): 7
L_CFP (max 15): 15

IEEE 802.15.4 WPAN Setting
Superframe Duration (SD): 245.76 ms
Beacon Interval (BI): 1966.08 ms
Time Slot Duration (TS): 15.36 ms
Number of routers: 7
Bandwidth per TS (R_TS): 0.390625 kbps
Duty Cycle: 12.5 %

Delay Bounds
Ddata (from end node to router): 3.42528 sec
Worst-case end-to-end delay (sum of per-hop delays): 14.8246 sec
Worst-case end-to-end delay (end-to-end service curve): 9.68916 sec

☐ show graphs

Figure 5.11: The GUI of the Matlab analytical model.

5 bytes at the data payload of every message. Hence, the maximum size of the MAC frame is equal to $MPDU_{max} = 192$ bits (i.e. MAC Header = 72 bits, FCS = 16 bits, Network Header = 64 bits, and Data Payload = 40 bits) assuming only destination PAN identifier and both source and destination addresses are present. Note that all nodes in the WSN have unique 16-bit short addresses assigned by the PAN Coordinator during the association process.

TinyOS 1.x flushes the reception buffer of the radio transceiver after processing the first arriving frame. Thus, the frames that arrive during the processing time of the first frame are discarded. This problem has been already reported and fixed in TinyOS 2.x. Since the proposed implementation of IEEE 802.15.4/ZigBee protocol stack was built over TinyOS 1.x, the aforementioned problem is overcame by setting the inter-frame spacing (IFS) time (i.e. time between two consecutive frames) such that no frame arrives during the frame processing time. A value of IFS equal to 3.07 ms was measured.

According to Eq. (5.43), the bandwidth guaranteed by one time slot for $SO = 4$ is equal to 3.125 kbps with 100% duty-cycle. Hence, in the experimental scenario with a 12.5% duty-cycle (i.e. $BO = BO_{min} = 7$), the guaranteed bandwidth of one time slot is equal to $R_{TS} = 3.125 \cdot 0.125 = 0.3906$ kbps. Let us assume $N_{end-node}^{TS} = 1$. Then, according to Eqs. (5.46) and (5.47), the maximum arrival rates of the sensory data are obtained as follows:

- $r_{data}^{MAX} = 456$ bps for $H_{sink} = 2$
- $r_{data}^{MAX} = 684$ bps for $H_{sink} = 1$
- $r_{data}^{MAX} = 911$ bps for $H_{sink} = 0$ (root)

As a result of $r_{data} \leq \min(r_{data}^{MAX})$ and $r_{data} \leq R_{TS}$, an average arrival rate equal to $r_{data} = 390$ bps, which corresponds to 4 frames (192 bits each) generated during one Beacon Interval (BI = 1.96608 sec), is considered. The burst tolerance is assumed to be equal to $b_{data} = 576$ bits, which corresponds to 3 frames generated at once. Hence, each sensor node transmits sensory data bounded by the arrival curve $\alpha_{data} = 576 + 390 \cdot t$. Note that Network Calculus based analytical model is bit-oriented, which means that sensory data are handled as a continuous bit stream with data rate r_{data}, while the experimental test-bed is frame-oriented, where data traffic is organized in frames of a given size. The frames can be generated at constant bit rate (CBR) or variable bit rate (VBR), but all data traffic must be upper bounded by the arrival curve α_{data} (Figure 5.10b).

Finally, let us summarize the complete network setting:

$N_{router}^{MAX} = 2$

$N_{end-node}^{MAX} = 1$

$H = 2$

$SO = 4$ (SD = 245.76 ms)

$BO = 7$ (BI = 1966.08 ms)

$MPDU_{max} = 192$ bits

$r_{data} = 390$ bps

$b_{data} = 576$ bits

IFS = 3.07 ms

$L_{CFP} = 15$

$\omega = 0$

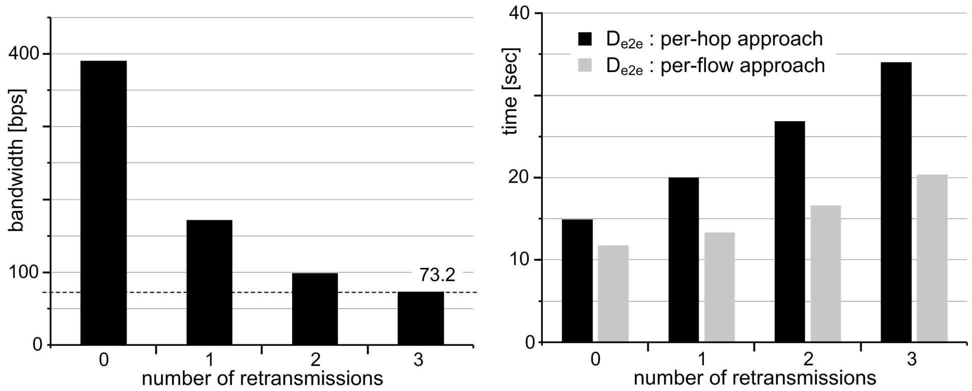
$macMaxFrameRetries = 0$

It is assumed the non-overlapping worst-case TDCS given by the following sequence of active portions of clusters 11, 01, 12, 24, 23, 21, 22. Note that the unacknowledged transmissions is only assumed.

5.8.2 Analytical evaluation

Number of retransmissions vs. timing performance

The unreliable and time-varying characteristics of wireless channels can be minimized using the acknowledgement and retransmission mechanisms. On the other side, each retransmission decreases guaranteed bandwidth and increases communication delay as depicted in Figure 5.12. Figure 5.12a



(a) The guaranteed bandwidth of 1 time slot.

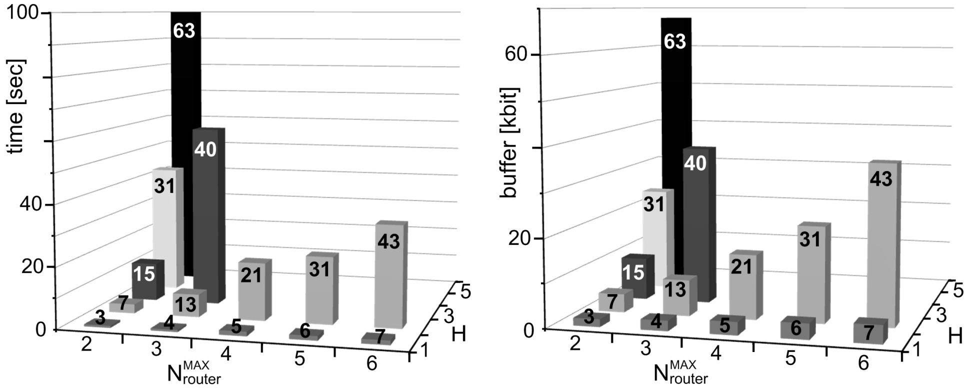
(b) The worst-case end-to-end delay.

Figure 5.12: The worst-case delay and bandwidth as a function of the number of retransmissions.

shows the guaranteed bandwidth of one time slot and Figure 5.12b the theoretical worst-case end-to-end delay as a function of the number of retransmissions (parameter *macMaxFrameRetries*) for $H_{sink} = 0$. The guaranteed bandwidth of one GTS time slot (Figure 5.12a) is obtained using Eq. (5.43) multiplied by the duty-cycle, which is equal to 12.5%. It can be observed that the minimum guaranteed bandwidth of one time slot is equal to 73.2 bps when three retransmissions are enabled. To obtain comparable end-to-end delays, the same number of time slots must be allocated to each end-node when consider different number of retransmissions. Hence, the average arrival rate of sensory data must be reduced to $r_{data} = 73$ bps. The other network settings are the same as mentioned in Section 5.8.1. The worst-case end-to-end delays obtained by per-flow approach introduces less pesimism than the per-hop approach, and end-to-end delays increase with the number of retransnmissions as shown in Figure 5.12b. These resuls confirm the previous assumptions. Acknowledged transmission with one eventual retransmission enlarges end-to-end delay by 13.5% against an unacknowledged transmission with no retranmissions, but also increases the reliability.

Network planning

The proposed methodology can be used for the planning of the cluster-tree topology as well. Let us consider the example of a convergecast application gathering sensory data at the root (i.e. $H_{sink} = 0$) and using the network settings as mentioned in Section 5.8.1. However, in this case, the largest



(a) The worst-case end-to-end delay. (b) The buffer requirement of the sink router.

Figure 5.13: The worst-case delay and buffer requirement as a function of N_{router}^{MAX} and H.

feasible configuration of the worst-case cluster-tree topology is achieved for $N_{router}^{MAX} = 2$ and $H = 2$. This means that a feasible worst-case cluster-tree topology given by the parameters N_{router}^{MAX} and H satisfies the network constraints given by the other parameters, namely r_{data}, b_{data}, SO, BO, $MPDU_{max}$, IFS, L_{CFP}, ω, $macMaxFrameRetries$ and $N_{end-node}^{MAX}$.

To obtain more illustrative results, the length of the IFS is reduced to the minimum value defined by 802.15.4 standard (see Section 5.7.1), $r_{data} =$ 25 bps, $SO = 2$, and keep the other settings. Figure 5.13a presents the worst-case end-to-end delay and Figure 5.13b buffer requirement of the sink router as a function of the height of the tree H and the maximum number of child routers N_{router}^{MAX}. In other words, Figure 5.13 presents all feasible configurations of the worst-case cluster-tree topology, which satisfy a given network constraints. The numerical values at the columns represent the total number of routers (N_{router}^{TOTAL}, Eq. (5.36)) in the network. It can be observed that there can be more feasible configurations for the same number of routers. For instance, the total number of 31 routers can be achieved with two configurations, namely $H = 2$ and $N_{router}^{MAX} = 5$ or $H = 4$ and $N_{router}^{MAX} = 2$. The buffer requirements at the sink router are almost the same for both configurations, but the first configuration provides around half of the worst-case end-to-end delay ($D_{e2e} = 22.18$ sec) compared with the second configuration ($D_{e2e} = 43.15$ sec). On the other side, the cluster-topology using the second configuration can spread out over a larger area due to the higher height H. So the system designer must find a trade-off for a given application-specific implementation.

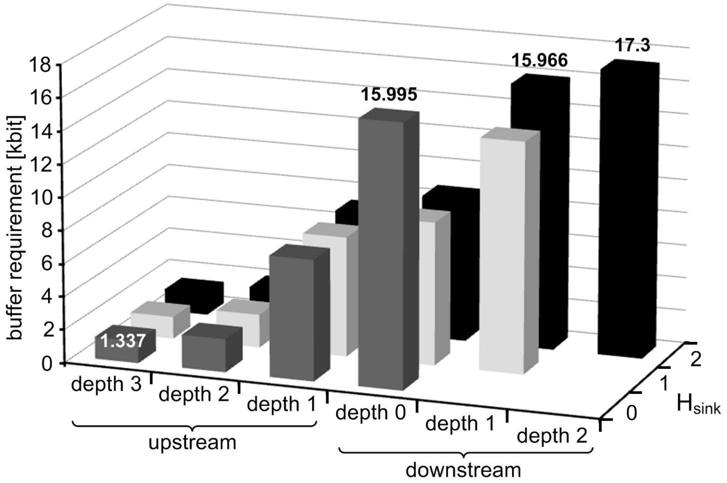
5.8.3 Experimental evaluation

Buffer requirements

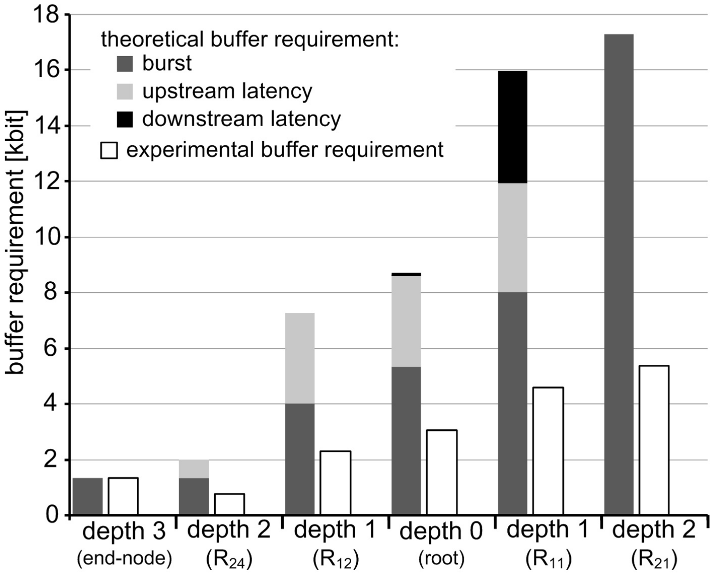
Figure 5.14a presents the theoretical worst-case buffer requirement of the routers at given depths and as a function of the sink position. It can be observed that end-nodes have the smallest buffer requirement as they are the leaves of the tree, and that the buffer requirement grows in the direction of the sink router. Since the sink can be associate to any router in a WSN and in order to avoid buffer overflow, all routers at depth i should allocate a buffer of capacity equal at least to the maximum buffer requirement at a given depth i (e.g. all routers at depth 0 allocate a buffer of capacity equal to 15.995 kbits), which effectively demonstrates how these analytical results can be used by a system designer.

Figure 5.14b shows the theoretical worst-case buffer requirements compared with the maximum values obtained through real experimentation, for $H_{sink} = 2$. First, the theoretical buffer requirements are divided into three portions according their origin, as was shown in Section 5.6.1. Observe that the cumulative effect of the burst is more important than the cumulative effect of the service latencies. The effect of the service latencies may be more important for other settings of b_{data} and r_{data}. So, the different settings of the sensory arrival curve affect the buffer requirements. The minor effect of the upstream service latency at depth 0 is given by the priority rules (Section 4.4.1), such that the data arriving during the transmit GTS (i.e. over the upstream link) are stored in the root until the receive GTS (i.e. downstream link), at the end of the same SD, is active and data is dispatched (Figure 5.9).

The next observation confirms that the theoretical values upper bound the experimental values. The pessimism of the theoretical bounds is justified by the fact that the Network Calculus analytical model is based on a continuous approach (arrival and service curves are continuous) in contrast to the real stepwise behaviour of data traffic and services (in the test-bed). In practice, the data is actually transmitted only during its GTS, while in the analytical model a continuous data flow during the whole BI is considered, since it represents the average rate and not the instantaneous rate. Figure 5.15 illustrates the problem and shows the arrival and service curves of a sensory data sent by an end-node to its parent router. The burst of the outgoing data b^{*}_{data} (Eq. (5.11)) is equal to Q^{TH}_{max}, in case of the analytical model, or Q^{EXP}_{max}, in the experimental case. Due to the cumulative flow effect, the differences between theoretical (Q^{TH}_{max}) and experimental (Q^{EXP}_{max}) values of buffer requirement grow with depth. The rate-latency service curve used



(a) The worst-case buffer requirement per router as a function of the depth and sink position.



(b) The theoretical vs. experimental buffer requirements.

Figure 5.14: The worst-case buffer requirement.

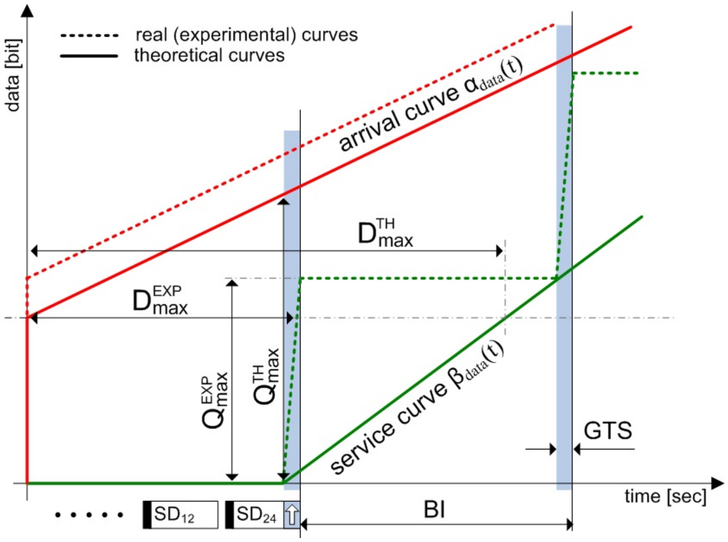


Figure 5.15: Theoretical vs. experimental data traffic (related to Figure 5.2).

in the analysis results from a trade-off between computing complexity and pessimism.

The numerical values of theoretical worst-case as well as experimental maximum buffer requirements are summarized in Table 5.1. The bandwidth requirements given by Eqs. (5.34) and (5.35), and the corresponding number of time slots are also presented. In Tables 5.1 and 5.2, U means an upstream router at depth i or an upstream link to a router at depth i, and D means a downstream router or a downstream link from a router at depth i.

Delay bounds

Figure 5.16 compares the worst-case, maximum and average values of per-hop delays bound in each router, and the end-to-end delay bounds for $H_{sink} = 2$. A first observation confirms that theoretical results upper bound the experimental results. The difference in theoretical worst-case (D^{TH}_{max}) and experimental maximum (D^{EXP}_{max}) delays (Figure 5.15) is given by the aforementioned continuous and stepwise behaviours of the analytical model and test-bed, respectively. The experimental delays comprise mainly the service latencies (Figure 5.15) decreasing in the direction of the sink (Figure 5.9). Hence, the maximum per-hop delays also decrease in the direction of the sink, as can be observed in Figure 5.16. The reduced downstream delay at depth 0 results from the priority rule (Section 4.4.1). The end-to-end delays bounds are quite high, even though the b_{data} and r_{data}

	depth		theoretical results (worst-case values)			experimental results (maximum values)
			R_i [kbps]	N_i^{TS}	Q_i [kbit]	Q_i [kbit]
$H_{sink} = 0$	0	U	1.7	3	**15.995**	5.376
	1	U	0.39	1	7.329	2.304
	2	U	–	–	2.008	0.768
$H_{sink} = 1$	0	D	1.56	4	8.667	3.072
		U	1.17	3	–	–
	1	D	–	–	14.02	5.376
		U	0.39	1	7.257	2.304
	2	U	–	–	2.008	0.768
$H_{sink} = 2$	0	D	1.56	4	8.667	3.072
		U	1.17	3	–	–
	1	D	2.34	6	**15.966**	4.608
		U	0.39	1	7.257	2.304
	2	D	–	–	**17.3**	5.376
		U	–	–	2.008	0.768
end-node			0.39	1	**1.344**	1.337

Table 5.1: Buffer requirements: theoretical vs. experimental results.

are low. This is mainly due to high value of $SO = 4$ (i.e. BI = 1.966 sec). Hence, the end-to-end delay bounds can be reduced using lower values of SO or higher bandwidth guarantees, using lower IFS, for example. Observe also that the worst-case end-to-end delay obtained by the per-flow approach introduces less pessimism than the delay from the per-hop approach (roughly by 50% smaller: 27.13 s $\rightarrow$ 13.65 s, as presented in Table 5.2).

Table 5.2 presents the worst-case, maximum and average numerical values of per-hop and per-flow delay bounds, and the end-to-end delays for different sink positions. Note that the average values were computed from a set of 15 runs, involving the transmission of 1155 frames each. The theoretical worst-case end-to-end delays are obtained as the sum of per-hop delays using Eq. (5.39), or by per-flow approach, which results in the family of service curves as a function of $\Theta \geq 0$. This analysis assumes $\Theta = T + (b_2/R)$ as a trade-off between computation complexity and optimality. The determination of the optimal service curve, leading to the lowest worst-case delay, will be addressed in future work.

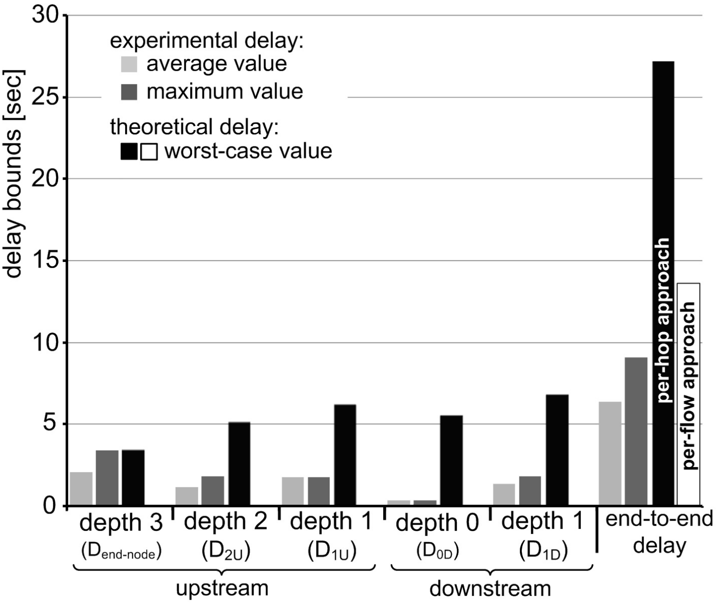


Figure 5.16: The theoretical vs. experimental delay bounds.

	depth	theoretical results (worst-case values)	experimental results	
			maximum	average
		D_i [sec]	D_i [sec]	D_i [sec]
$H_{sink} = 0$	1 U	6.257	1.764	1.308
	2 U	5.143	1.812	1.602
	D_{e2e}	14.82/**9.69**	**7.154**	4.952
$H_{sink} = 1$	0 D	5.547	0.104	0.099
	1 U	6.195	1.76	1.728
	2 U	5.143	1.809	1.602
	D_{e2e}	20.31/**10.53**	**7.251**	5.471
$H_{sink} = 2$	0 D	5.547	0.104	0.099
	1 D	6.814	1.812	1.321
	1 U	6.195	1.766	1.728
	2 U	5.143	1.814	1.135
	D_{e2e}	27.13/**13.65**	**9.074**	6.325
end-node D_{data}		3.425	3.402	2.042

Table 5.2: Delay bounds: theoretical vs. experimental results.

Duty-cycle vs. timing performance

In Section 5.7.3 was mentioned that to maximize the lifetime of a WSN, low duty-cycles are required. On the other hand, low duty-cycles enlarge the timing performance of a WSN. These assumptions were confirmed as depicted in Figure 5.17, which shows the theoretical worst-case and experimental maximum end-to-end delays as a function of duty-cycle for $H_{sink} = 0$. The value of SO is set to 4 and the decreasing duty-cycles are obtained by increasing BO. Note that for $SO = 4$, the minimum BO is equal to 7. To avoid the lack of bandwidth for smaller duty-cycles, the average arrival rate must be reduce to $r_{data} = 0.190$ kbps (note that $r_{data}^{MAX} = 0.195$ kbps for the smallest duty-cycle equal to 3.125%). The other network settings are the same as in previous experiments. The theoretical worst-case end-to-end delays are obtained by per-hop and per-flow approaches (Section 5.6.2). The observation again confirms that the theoretical results upper bound the experimental results, and the worst-case delay obtained by the per-flow approach offers less pessimism than the delay from the per-hop approach.

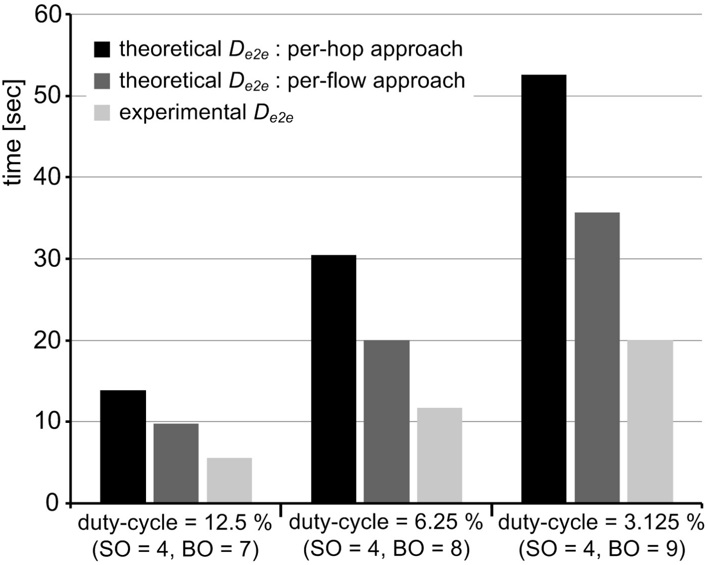


Figure 5.17: The theoretical worst-case and experimental maximum end-to-end delays as a function of duty-cycle for $H_{sink} = 0$.

5.9 Discussion on mobility support

This section outlines several issues related to how to support the sink mobility in a cluster-tree WSN.

5.9.1 Adapting the logical topology

All network nodes (i.e. routers and end-nodes) are assumed to be static; note that even the sink router is static - only the sink entity can be mobile. Considering the mobile sink behaviour, the logical topology has to be adapted to allow sensory data to reach the new sink location. Two approaches to mobile sink behaviour are assumed: *topology update* (Figure 5.18a), applicable to the cases where the sink moves more frequently (i.e. in a continuous fashion), and *topology rebuilding* (Figure 5.18b), applicable to the cases where the sink changes location less frequently (sporadically). Generally, the topology update approach should be chosen when the network inaccessibility time resulting from the topology rebuilding is greater than the maximum time between two consecutive sink movements (i.e. two consecutive sink associations with different sink routers).

The topology update

When the sink moves very frequently, it might be more adequate to keep the logical topology unchanged (the same original root and network deployment). Hence, the path of data flows in the direction of the sink has to be updated accordingly (dubbed *topology update*). The length of this updated path is equal to the number of hops between the previous and current sink router positions. In the worst-case, when the sink moves between two farthest routers of different sub-trees of the root, the length of the update path becomes twice the height of the cluster-tree. Thus, the network inaccessibility time will depend on the length of the update path and on the TDCS. In case of the worst-case TDCS, the inaccessibility time is expressed as the period of the WC-TDCS multiplied by the length of the update path. This inaccessibility time is much smaller than the network inaccessibility time resulting from the topology rebuilding (see next), and also the energy consumption during the topology update is expected to be lower. On the other hand, the resource requirements (i.e. the resources guaranteed along the longest data path) in the updated topology are higher as compared to the rebuilt topology. It results in a balanced topology with unbalanced load requiring higher energy consumption and higher end-to-end delay bounds.

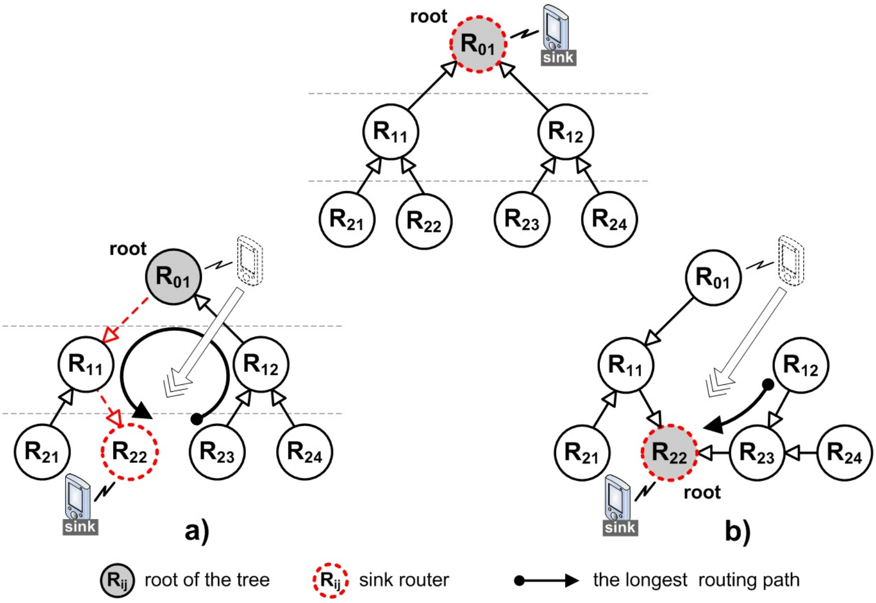


Figure 5.18: Two approaches to mobile sink behaviour: a) topology update; b) topology rebuilding.

The topology rebuilding

When the sink moves infrequently, it might be more adequate to rebuild the logical topology from scratch, according to the current position of the sink (dubbed *topology rebuilding*). The current sink router becomes the root of the tree and initiates the logical topology rebuilding of the cluster-tree WSN. The rebuilt cluster-tree topology may keep the same worst-case parameters or exceed some of them (e.g. the height of the tree). There can be more than one feasible cluster-tree topology. Consequently, this topology rebuilding procedure introduces higher network inaccessibility times and consumes more energy as compare to the previous approach. On the other hand, the run time resource requirements in this approach are lower as compared to the updated topology, resulting in reduced delay and buffer bounds. It results in a balanced topology with balanced load requiring lower energy consumption and lower end-to-end delay bounds.

5.9.2 Routing protocol

Although it is not a main focus of this work, in this section a simple routing protocol is proposed. For comparing purpose, let us consider the example of the tree routing protocol described in the ZigBee specification [10]. It relies on a distributed address assignment mechanism that provides a finite sub-block of hierarchical unique network addresses to each parent router. Since the tree routing algorithm is address-based, each node requires assignment of a unique network address from its parent router. If a parent router lacks available addresses, the address assignment mechanism imposes address redistribution, causing some network inaccessibility time. Conversely, the proposed routing protocol reduces the inaccessibility time and simplifies the joining of new nodes. The messages are routed locally, from cluster to cluster, until reaching the sink. Thus, the node's address must be unique only between the adjacent clusters, and a joining node is assigned an arbitrary unused address. On the other hand, the proposed protocol is only suitable for a cluster-tree WSN where all data flows are routed to the sink.

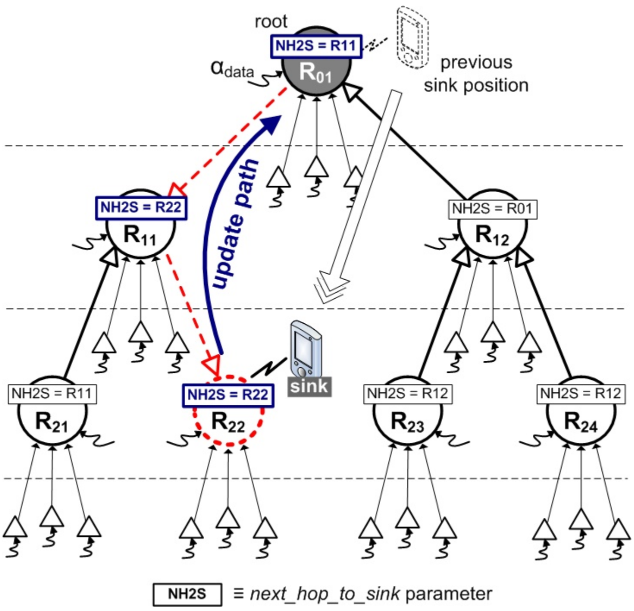
Routing algorithm

The algorithm assumes that each router is assigned an address, prior to run-time, which is unique with respect to its adjacent clusters. New router's parameter, called *next_hop_to_sink* (referred to as $NH2S$ hereafter), is defined. The value of this parameter is equal to the address of the next router on the path towards the current sink location; all incoming traffic is forwarded to this direction. In case of the sink router, this parameter is equal to its own address. After the cluster-tree topology is built up (or rebuilt in case of the topology rebuilding approach), the $NH2S$ parameter of each router contains the address of its parent router (in the cluster-tree hierarchy) or its own address in case of root, and the sink is initially associated to the root.

Route update algorithm

In case of the topology update approach, the path of data flows in the direction of the sink has to be updated, to enable data flows to be correctly routed to the sink. The *route update algorithm* is proposed as a part of the routing protocol that tracks the frequently moving sink and updates the $NH2S$ parameters of the relevant routers.

This algorithm is illustrated in Figure 5.19 and is described next. Let us assume that the sink has moved (from R_{01}, in Figure 5.19a) and associated to another router (R_{22}). The current sink router must notify all routers along the update path to the previous sink router position (R_{01}). The routers along the update path (R_{22}, R_{11} and R_{01}) update their $NH2S$ parameters. On the contrary, the routers out of the update path keep the same $NH2S$



(a) The illustration of the route update algorithm.

Route Update Algorithm

```
if I am current sink router
  send update_path(SrcAddr = my_address, DestAddr = NH2S)
  NH2S = my_address
else
  wait for an update_path
  int src_address = update_path(SrcAddr)

  if NH2S != my_address
  /* I am not the former sink router,
          i.e.  the end of the update path */
    send update_path(SrcAddr = my_address, DestAddr = NH2S)
  end

  NH2S = src_address
end
```

(b) Pseudo-code for the route update algorithm running at each router.

Figure 5.19: The route update algorithm.

values. First, the current sink router (R_{22}) sends an *update_path* message to its *next_hop_to_sink* router (R_{11}) and then updates its $NH2S$ parameter with its own address. After receiving this *update_path* message, the next router (R_{11}) on the update path sends another *update_path* message to its *next_hop_to_sink* router (R_{01}) and then updates its $NH2S$ parameter with the source address embedded in the received *update_path* message (R_{22}). This procedure is repeated router by router until reaching the former sink router (i.e. a router which $NH2S$ parameter is equal to its own address - R_{01}). The former sink router (R_{01}) does not send a new *update_path* message and only updates its $NH2S$ variable with the source address of the received *update_path* message.

During the update process, just some links between correspondent routers are reversed, thus not impacting the entire network (involving the minimum number of routers/messages), so normal network operation can quickly be resumed. As a result, this algorithm requires a minimum amount of control-related traffic and reduces network inaccessibility times (the computation of these inaccessibility times is out of the context of this work).

5.10 Conclusions

Modelling the fundamental performance limits of Wireless Sensor Networks (WSNs) is of paramount importance to understand their behaviour under the worst-case conditions and to make the appropriate design choices. In that direction this chapter contributes with a methodology based on Network Calculus, which enables quick and efficient worst-case analysis and dimensioning of static or even dynamically changing cluster-tree WSNs where the data sink can either be static or mobile, i.e. can be associated to any router in the WSN. The proposed analytical methodology (closed-form recurrent expressions) enables to guarantee the routers' buffer size to avoid buffer overflows and to minimize clusters' duty-cycle (maximizing nodes' lifetime) still satisfying that messages' deadlines are met.

The chapter also shows how to instantiate the generic methodology in IEEE 802.15.4/ZigBee, which are promising technologies for WSN applications. Finally, based on the Commercial-Off-The-Shelf (COTS) technologies, namely TelosB motes [14] running open-ZB protocol stack [71] over TinyOS [72], a multiple cluster test-bed has been configured. This test-bed enabled to assess the validity and pessimism of the worst-case theoretical results (buffer requirements and message end-to-end delays), by comparing these to the maximum and average values measured in the experiments. The results confirm that the theoretical values upper bound the experimental values, the buffer requirements grow in the direction of the sink router, and the end-to-end delays increase with number of retransmissions. The observation also shows that the worst-case end-to-end delay obtained by the per-flow approach offers less pessimism than the delay from the per-hop approach (roughly by 50%). Using the proposed methodology for the planning of the cluster-tree networks is demonstrated by the illustrative example.

The future work includes improving the current methodology to encompass clusters operating at different duty-cycles and to provide a model that enables real-time control actions, i.e. the sink assuming the role of controlling sensor/actuator nodes.

APPENDIX

5.A Table of symbols

The following table reports the symbols that are used through the Chapter 5, along with their definition.

Symbol	Definition
$R(t)$	input cumulative function
$R^*(t)$	output cumulative function
$\alpha(t)$	arrival curve
$\beta(t)$	guaranteed service curve
$\alpha^*(t)$	output bound constraining the output function $R^*(t)$
D_{max}	delay bound
Q_{max}	backlog bound
$\odot$	min-plus deconvolution
$\otimes$	min-plus convolution
$\beta_1^{eq}(t,\Theta)$	equivalent service curve for flow 1
$1_{\{expr\}}$	$1_{\{expr\}}$ is equal to 1 if *expr* is true, and 0 otherwise.
$(x)^+$	$(x)^+ = \max(0,x)$
$\alpha_{b,r}(t)$	affine arrival curve with rate r and burst size b

$\beta_{R,T}(t)$	rate-latency service curve with rate R and latency T
H	height of the tree
$N_{end-node}^{MAX}$	maximum number of child end-nodes
N_{router}^{MAX}	maximum number of child routers
H_{sink}	maximum depth of the sink router
$\beta_{end-node}(t)$	rate-latency service curve guaranteed to end-nodes
$\alpha_{data}(t)$	affine arrival curve constraining sensory data
$\bar{\alpha}_i(t)$	affine arrival curve constraining the input function $R(t)$ of a router at a depth i
$\alpha_i^*(t)$	output bound constraining the output function $R^*(t)$ of a router at a depth i
$\beta_i(t)$	rate-latency service curve guaranteed by a router at depth i
ω	binary variable which is equal to 1 if routers have sensing capabilities; otherwise ω is equal to 0
Q_{iU}	the required buffer size of an upstream router at a depth i
D_{iU}	delay bound between a child router at depth i and its parent router at depth $i-1$
D_{iD}	delay bound between a parent router at depth i and its child router at depth $i+1$
D_{e2e}	the worst-case end-to-end delay

Table 5.3: Table of symbols.

Chapter 6

Conclusions

This thesis addresses several problems and open research issues in the area of design and performance evaluation of the cluster-tree Wireless Sensor Networks (WSN) focusing on the energy efficient and real-time behaviour. The proposed methodologies are instantiated in the IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs, as an illustrative example that confirms the applicability of general approach to specific protocols. The IEEE 802.15.4/ZigBee protocols, which are the leading technologies for WSNs, have been chosen for their ability to provide predictable QoS guarantees for the time-sensitive and energy efficient wireless sensor applications.

Since the simulation is important approach to developing and evaluating the systems, the Opnet simulation model of the IEEE 802.15.4/ZigBee protocols has been proposed in Chapter 3. This chapter also provides the performance evaluation of the GTS mechanism, comparing the obtained simulation results with the ones that were previously obtained using an analytical model based on Network Calculus. The behaviours of both models are roughly identical in terms of the GTS data throughput and the media access delay. An optimal setting of the IEEE 802.15.4 GTS mechanism for obtaining maximum data throughput and minimum delay has also been proposed. For applications with low data arrival rates and low buffer capacities, the maximum utilization of the allocated GTS is achieved for low SOs (3–4). However, the SO equal to 2 is the most suitable value for providing real-time guarantees in time-sensitive WSNs, since it grants the minimum access delay for the GTS frames. High SOs are not suitable for ensuring efficient usage of the GTS neither in terms of data throughput nor media access delay.

The Chapter 4 shows how to minimize the energy consumption of the nodes by setting the beacon interval (TDCS period) as long as possible while respecting all parameters of data flows and avoiding possible inter-cluster collisions in a static cluster-tree WSN with predefined set of time-bounded data flows. Since the problem is formulated as a cyclic extension of RCPS/TC, the users are not restricted to a particular implementation but they can make a similar extension to any of the algorithms solving this problem. In this chapter, the solution is shown on iterative calls of the ILP algorithm, which gives acceptable performance for static configuration of middle-sized WSNs.

Modelling the fundamental performance limits of WSNs is of paramount importance to understand their behaviour under the worst-case conditions and to make the appropriate design choices. In that direction the Chapter 5 contributes with a methodology based on Network Calculus, which enables quick and efficient worst-case analysis and dimensioning of static or even dynamically changing cluster-tree WSNs where a static or mobile sink gathers data from all sensor nodes. The proposed analytical methodology (closed-form recurrent expressions) enables to guarantee the routers' buffer size to avoid buffer overflows and to minimize clusters' duty-cycle (maximizing nodes' lifetime) still satisfying that messages' deadlines are met. The experimental study assess the validity and pessimism of proposed methodology and shows how this methodology can be used for the planning of the cluster-tree networks. The results confirm that the theoretical values upper bound the experimental values, and the buffer requirements grow in the direction of the sink router. The observation also shows that the worst-case end-to-end delay obtained by the per-flow approach offers less pessimism than the delay from the per-hop approach (roughly by 50%).

The communication errors such as message corruption or message loss come from unreliable and time-varying characteristics of wireless channels. To increase the reliability of data transmission, the acknowledgement and retransmission mechanisms are employed. On the other side, the simulation and experimental results performed in this thesis demonstrate that each retransmission also increases the energy consumption of the nodes and the end-to-end communication delay. Hence, the interdependence of reliability, energy consumption and timeliness make the network design more complex.

Using the proposed methodologies and simulation model, system designers are able to configure the IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs and easily find the trade-off between the reliability of data transmission, the energy consumption of the nodes and the end-to-end communication delay for a given application-specific implementation prior to the network deployment.

Bibliography

[1] J. Stankovic, I. Lee, A. Mok, and R. Rajkumar, “Opportunities and Obligations for Physical Computing Systems,” *IEEE Computer journal*, vol. 38, no. 11, pp. 25–33, Nov. 2005.

[2] T. He, S. Krishnamurthy, J. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, and L. Gu, “Energy-efficient surveillance system using wireless sensor networks,” in *Proc. of the 2nd International Conf. on Mobile systems, Applications, and Services (MobiSys)*, Jun. 2004.

[3] T. He, P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J. Stankovic, and T. Abdelzaher, “Achieving Real-Time Target Tracking Using Wireless Sensor Networks,” in *Proc. of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Apr. 2006.

[4] M. Kohvakka, M. Hannikainen, and T. Hamalainen, “Wireless sensor network implementation for industrial linear position metering,” in *Proc. of the 8th Euromicro Conference on Digital System Design (DSD)*, Sep. 2005, pp. 267–275.

[5] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher, “A Spatiotemporal Communication Protocol for Wireless Sensor Networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 10, pp. 995–1006, Oct. 2005.

[6] B. Raman and K. Chebrolu, “Censor networks: a critique of ”sensor networks” from a systems perspective,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 75–78, Jul. 2008.

[7] J. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou, “Real-Time Communication and coordination in Embedded Sensor Networks,” *Proceedings of the IEEE*, vol. 91, no. 7, pp. 1002–1022, Jul. 2003.

[8] H. Bai and M. Atiquzzaman, "Error modeling schemes for fading channels in wireless communications: A survey," *IEEE Communications Surveys and Tutorials*, vol. 5, no. 2, pp. 2–9, Oct. 2003.

[9] *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE SA Standards Board Std. 802.15.4, 2006.

[10] *ZigBee Specification*, ZigBee Standards Organization Std. 053 474r13, 2006.

[11] A. Koubâa, M. Alves, and E. Tovar, "GTS Allocation Analysis in IEEE 802.15.4 for Real Time Wireless Sensor Networks," in *Proc. of the 14th Workshop on Parallel and Distributed Real Time Systems (WPDRTS)*, Apr. 2006.

[12] In-Stat. (2009) Automatic Meter Reading (AMR) and Smart Energy To Be the Winning Application for 802.15.4/ZigBee. [Online]. Available: http://www.instat.com

[13] *Part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)*, IEEE SA Standards Board Std. 802.15.4, 2003.

[14] Crossbow Technology, Inc. (2009) TelosB Mote Datasheet. [Online]. Available: http://www.xbow.com

[15] A. Koubâa, M. Alves, E. Tovar, and A. Cunha, "An implicit GTS allocation mechanism in IEEE 802.15.4 for time-sensitive wireless sensor networks: theory and practice," *Real-Time Systems Journal*, vol. 39, no. 1–3, pp. 169–204, Aug. 2008.

[16] C. P. Singh, O. P. Vyas, and M. K. Tiwari, "A Survey of Simulation in Sensor Networks," in *Proc. of the 4th International Conf. on Computational Intelligence for Modelling Control and Automation (CIMCA)*, Dec. 2008, pp. 867–872.

[17] OPNET Technologies, Inc. (2009) The Opnet Modeler network simulator version 15.0.A. [Online]. Available: http://www.opnet.com

[18] S. Khazzam. (2005) IEEE 802.15.4 MAC Protocol Model (used in ZigBee low-rate WPAN). [Online]. Available: http://www.opnet.com

[19] K. Fall and K. Varadhan. (2009) The Network Simulator 2 (ns-2). [Online]. Available: http://www.isi.edu/nsnam/ns

[20] A. Koubâa, M. Alves, B. Nefzi, and Y. Song, “Improving the IEEE 802.15.4 Slotted CSMA/CA MAC for Time-Critical Events in Wireless Sensor Networks,” in *Proc. of the 5th Workshop on Real Time Networks (RTN)*, Jul. 2006.

[21] A. Varga. (2009) OMNeT++ 4.0 network simulator. [Online]. Available: http://www.omnetpp.org

[22] J. Zheng and M. J. Lee, “Comprehensive Performance Study of IEEE 802.15.4,” in *Sensor Network Operations*, S. Phoha, T. L. Porta, and C. Griffin, Eds. New York, USA: Wiley-IEEE Press, 2006, ch. 4, pp. 218–237.

[23] F. Chen and F. Dressler, “A Simulation Model of IEEE 802.15.4 in OMNeT++,” in *Proc. of the 6th GI/ITG KuVS Fachgesprach Drahtlose Sensornetze (FGSN)*, Jul. 2007, pp. 35–38.

[24] F. Chen, N. Wang, R. German, and F. Dressler, “Performance Evaluation of IEEE 802.15.4 LR-WPAN for Industrial Applications,” in *Proc. of the 5th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS)*, Jan. 2008, pp. 89–96.

[25] J. Hurtado-Lopez, E. Casilari, and A. Ariza-Quintana, “Enabling IEEE 802.15.4 cluster-tree topologies in OMNeT++,” in *Proc. of the 2nd International Conference on Simulation Tools and Techniques (SIMUTools)*, Mar. 2009.

[26] A. Koubâa, M. Alves, and E.Tovar, “A Comprehensive Simulation Study of Slotted CSMA/CA for IEEE 802.15.4 Wireless Sensor Networks,” in *Proc. of the 5th Workshop on Factory Communication Systems (WFCS)*, Jun. 2006, pp. 183–192.

[27] P. Jurčík and A. Koubâa. (2009) IEEE 802.15.4/ZigBee OPNET Simulation Model v3.0. [Online]. Available: http://www.open-zb.net

[28] Crossbow Technology, Inc. (2009) MICAz Mote Datasheet. [Online]. Available: http://www.xbow.com

[29] P. Jurčík and A. Koubâa, “The IEEE 802.15.4 OPNET Simulation Model: Reference Guide v2.0,” IPP-HURRAY Research Group, CISTER/ISEP, Polytechnic Institute of Porto, Portugal, Tech. Rep. TR-070509, May 2009.

[30] I. Akyildiz, S. Weilian, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[31] J. L. Boudec and P. Thiran, “Network Calculus: A Theory of Deterministic Queuing Systems for the Internet,” in *Lecture Notes in Computer Science*, G. Goos, J. Hartmanis, and J. van Leeuven, Eds. New York, USA: Springer-Verlag, 2004, vol. 2050.

[32] P. Havinga and S. Smit, “Energy-efficient TDMA medium access control protocol scheduling,” in *Proc. of the 1st Asian International Mobile Computing Conf. (AMOC)*, Oct. 2000.

[33] W. Ye, J. Heidemann, and D. Estrin, “An Energy-Efficient MAC protocol for Wireless Sensor Networks,” in *Proc. the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Jun. 2002, pp. 1567–1576.

[34] T. van Dam and K. Langendoen, “An adaptive energy-efficient MAC protocol for wireless sensor networks,” in *Proc. of the 1st international conference on Embedded networked sensor systems (SenSys)*, Nov. 2003, pp. 171 – 180.

[35] J. Polastre, J. Hill, and D. Culler, “Versatile Low Power Media Access for Wireless Sensor Networks,” in *Proc. of the 2nd international conference on Embedded networked sensor systems (SenSys)*, Nov. 2004, pp. 95 – 107.

[36] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher, “Real-time Power-Aware Routing in Sensor Networks,” in *Proc. of the 14th IEEE International Workshop on Quality of Service (IWQoS)*, Jun. 2006, pp. 83–92.

[37] J. Trdlička, M. Johansson, and Z. Hanzálek, “Optimal Flow Routing in Multi-hop Sensor Networks with Real-Time Constraints through Linear Programming,” in *Proc. of the 12th IEEE International Conf. on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2007, pp. 924–931.

[38] K. Akkaya and M. Younis, “An energy-aware QoS routing protocol for wireless sensor networks,” in *Proc. of the 23rd International Conf. on Distributed Computing Systems (ICDCS)*, May 2003, pp. 710–715.

[39] E. Toscano and L. LoBello, “A topology management protocol with bounded delay for wireless sensor networks,” in *Proc. of 13th International Conf. on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2008, pp. 942–951.

[40] L. LoBello and E. Toscano, “An Adaptive Approach to Topology Management in Large and Dense Real-Time Wireless Sensor Networks,” *IEEE Transaction on Industrial Informatics*, vol. 5, no. 3, pp. 314–324, Aug. 2009.

[41] 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*, vol. 40, no. 3, pp. 321–354, Oct. 2008.

[42] R. Holte, A. Mok, L. Rosier, I. Tulchinsky, and D. Varvel, “The pinwheel: A real-time scheduling problem,” in *Proc. of the 22nd Hawaii International Conf. on System Sciences (HICSS)*, Jan. 1989, pp. 693–702.

[43] R. Diestel, *Graph Theory.* New York, USA: Springer-Verlag, 2005.

[44] C. de M. Cordeiro and D. Agrawal, *Ad Hoc Sensor Networks: Theory and Applications.* World Scientific, 2006.

[45] K. Neumann, C. Schwindt, and J. Zimmermann, *Project Scheduling with Time Windows and Scarce Resources.* Springer, 2003.

[46] P. Brucker, A. Drex, R. Möhring, K. Neumann, and E. Pesch, “Resource-constrained project scheduling: Notation, classification, models, and methods,” *European Journal of Operational Research*, vol. 112, no. 1, pp. 3–41, 1999.

[47] B. de Dinechin, “Simplex scheduling: More than lifetime-sensitive instruction scheduling,” in *Proc. of the International Conf. on Parallel Architecture and Compiler Techniques*, 1994, pp. 327–330.

[48] R. Alur and D. Dill, “A theory of timed automata,” *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[49] Sid-Ahmed-Ali Touati and Z. Mathe, “Periodic register saturation in innermost loops,” *Journal of Parallel Computing*, vol. 35, no. 4, pp. 239–254, Apr. 2009.

[50] P. Jurčík and Z. Hanzálek. (2009) Matlab tool for TDCS. [Online]. Available: http://dce.felk.cvut.cz/hanzalek/TDCS

[51] A. Makhorin. (2009) GLPK (GNU Linear Programming Kit) 4.38. [Online]. Available: http://www.gnu.org/software/glpk

[52] Z. Hanzálek and P. Šůcha, "Time Symmetry of Project Scheduling with Time Windows and Take-give Resources," in *Proc. of the 4th Multidisciplinary International Scheduling Conf.: Theory and Applications (MISTA)*, Aug. 2009.

[53] Z. Hu and B. Li, "Fundamental Performance Limits of Wireless Sensor Networks," in *Ad Hoc and Sensor Networks*, Y. Xian and Y. Pan, Eds. New York, USA: Nova Science Publishers, 2004.

[54] T. Abdelzaher, S. Prabh, and R. Kiran, "On real-time capacity limits of multihop wireless sensor network," in *Proc. of the 25th IEEE International Real-Time Systems Symposium (RTSS)*, Dec. 2004, pp. 359–370.

[55] J. Gibson, G. Xie, and Y. Xiao, "Performance limits of fair-access in sensor networks with linear and selected grid topologies," in *Proc. of the 50th IEEE Global Communications Conf. (GLOBECOM)*, Nov. 2007, pp. 688–693.

[56] C. Lee, E. Ekici, and E. Felemban, "MMSPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 738–754, Jun. 2006.

[57] J. Schmitt and U. Roedig, "Sensor Network Calculus - A Framework for Worst Case Analysis," in *Proc. of the 1st IEEE/ACM Conf. on Distributed Computing in Sensor Systems (DCOSS)*, Jun. 2005, pp. 141–154.

[58] J. Schmitt, F. Zdarsky, and L. Thiele, "A Comprehensive Worst-Case Calculus for Wireless Sensor Networks with In-Network Processing," in *Proc. of the 28th IEEE Real-Time Systems Symposium (RTSS)*, Dec. 2007, pp. 193–202.

[59] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea, "Tight end-to-end per-flow delay bounds in FIFO multiplexing sink-tree networks," *Performance Evaluation*, vol. 63, no. 9, pp. 956–987, Oct. 2006.

[60] J. Schmitt and U. Roedig, “Worst Case Dimensioning of Wireless Sensor Networks under Uncertain Topologies,” in *Proc. of the 1st Workshop on Resource Allocation in Wireless NETworks (RAWNET)*, Apr. 2005.

[61] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo, “An Implicit Prioritized Access Protocol for Wireless Sensor Networks,” in *Proc. of the 23rd IEEE Real-Time Systems Symposium (RTSS)*, Dec. 2002, pp. 39–48.

[62] T. Facchinetti, L. Almeida, G. C. Buttazzo, and C. Marchini, “Real-time resource reservation protocol for wireless mobile ad hoc networks,” in *Proc. of the 25th IEEE International Real-Time Systems Symposium (RTSS)*, Dec. 2004, pp. 382–391.

[63] T. Crenshaw, S. Hoke, A. Tirumala, and M. Caccamo, “Robust implicit EDF: A wireless MAC protocol for collaborative real-time systems,” *ACM Transactions on Embedded Computing Systems*, vol. 6, no. 4, p. 28, Sep. 2007.

[64] S. Gandham, M. Dawande, R. Prahash, and S. Venkatesan, “Energy efficient schemes for wireless sensor networks with multiple mobile base stations,” in *Proc. of the 46th IEEE Global Communications Conf. (GLOBECOM)*, Dec. 2003, pp. 377–381.

[65] W. Poe and J. Schmitt, “Minimizing the Maximum Delay in Wireless Sensor Networks by Intelligent Sink Placement,” University of Kaiserslautern, Germany, Tech. Rep. 362/07, 2007.

[66] A. Koubâa, M. Alves, and E. Tovar, “Modeling and Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks: proofs and computation details,” IPP-HURRAY Research Group, CISTER/ISEP, Polytechnic Institute of Porto, Portugal, Tech. Rep. TR-060601, 2006.

[67] S. Prabh, “Real-Time Wireless Sensor Networks,” Ph.D. dissertation, Department of Computer Science, University of Virginia, VA, USA, May 2007.

[68] A. Koubâa, M. Alves, and E. Tovar, “Modeling and Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks,” in *Proc. of the 27th Real Time Systems Symposium (RTSS)*, Dec. 2006, pp. 412–421.

[69] A. Koubâa and Y. Song, “Evaluation and improvement of response time bounds for real-time applications under non-pre-emptive Fixed Priority Scheduling,” *International Journal of Production Research*, vol. 42, no. 14, pp. 2899–2913, Jul. 2004.

[70] A. Cunha, R. Severino, N. Pereira, A. Koubâa, and M. Alves, “ZigBee over TinyOS: implementation and experimental challenges,” in *Proc. of the 8th Portuguese Conf. on Automatic Control (CONTROLO)*, Jul. 2008, pp. 911–916.

[71] A. Cunha, A. Koubâa, R. Severino, and M. Alves, “Open-ZB: an open source implementation of the IEEE 802.15.4/ZigBee protocol stack on TinyOS,” in *Proc.of the 4th IEEE International Conf. on Mobile Ad-hoc and Sensor Systems (MASS)*, Oct. 2007.

[72] TinyOS Alliance. (2009) TinyOS open-source OS for wireless embedded sensor networks. [Online]. Available: http://www.tinyos.net

[73] P. Jurčík. (2009) Matlab Tool for the Worst-Case Dimensioning of IEEE 802.15.4/ZigBee Cluster-Tree WSNs. [Online]. Available: http://www.open-zb.net

[74] Chipcon. (2009) C2420DK development kit datasheet. [Online]. Available: http://www.ti.com

[75] Daintree Networks. (2009) Daintree sensor network analyzer (SNA). [Online]. Available: http://www.daintree.net

Index

best-effort, 10, 21, 22, 81

cluster-tree, 14, 16, 20, 41, 45, 48, 61, 79, 83, 84, 93, 98, 116

energy
  consumption, 48, 51, 116, 117

IEEE 802.15.4, 7, 10, 13, 20, 23, 48, 61, 98, 105

model
  analytical, 24, 28, 105, 107, 112
  simulation, 20, 21, 24, 61

Network Calculus
  arrival curve, 24, 74, 77
  backlog bound, 76
  delay bound, 75
  methodology, 24, 74
  service curve, 24, 75, 78
    network-wide, 96, 97

Opnet Modeler, 20

real-time, 21, 22, 34, 81

scheduling
  cyclic, 54

TDCS, *see* Time Division Cluster Scheduling
Time Division Cluster Scheduling, 48, 83, 84
TinyOS, 102, 105, 106

Wireless Sensor Network, 7, 41, 45, 48, 61, 79, 80, 84, 94
WSN, *see* Wireless Sensor Network

ZigBee, 7, 12, 13, 20, 21, 48, 79, 98, 105, 117

Curriculum vitae

Petr Jurčík was born in České Budějovice, Czech Republic, in 1978. He received his master of science degree in cybernetics and control engineering in Faculty of Electrical Engineering at Czech Technical University (CTU FEE) in Prague, Czech Republic in 2003. In 2002, he spent 7 months at the Institut für Automation und Kommunikation (ifak) in Magdeburg, Germany working on his master's thesis, which has been focused on the monitoring of the Profibus-PA fieldbus with Bluetooth wireless technology.

Since 2003 he has been working towards the doctor of philosophy degree (Ph.D.) in electrical engineering and information technology at the Department of Control Engineering, CTU FEE. From 2003 to 2005, he participated in the project of Control and Information System Designer (C&ISD) in cooperation with UNIS-Brno and DataPartner. His teaching activities at CTU FEE cover courses on Distributed control systems and .NET C# programming. He published and reviewed several conference papers (e.g. ETFA, RTCSA, MASCOTS, ECRTS) and journal papers (e.g. IEEE TII, IEEE TVT, IEEE TPDS, Elsevier Ad Hoc Networks, ACM TOSN). He also participated in the organization of the international conference ECRTS'08 and the international school SensorNets'09.

At Universidad Politécnica de Madrid, Spain, he participated in the project concerning over-the-air programming in wireless networks for 2 months in 2004. Petr also worked for ST Microelectronics in Prague, Czech Republic as a part-time software engineer for 10 months in 2006. Then, he joined the CISTER/IPP-HURRAY Research Unit in Porto, Portugal in 2007. For 2 years, he worked as a research associate in the area of real-time analysis and dimensioning of cluster-tree sensor networks.

Author's publications

Publications in journals

- P. Jurčík, R. Severino, A. Koubâa, M. Alves, and E. Tovar.: Dimensioning and Worst-case Analysis of Cluster-Tree Sensor Networks. In review for *ACM Transactions on Sensor Networks.*
- 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. In review for *IEEE Transactions on Industrial Informatics.*

Publications in conference proceedings

- P. Jurčík and Z. Hanzálek.: Construction of the Bounded Application-layer Multicast Tree in the Overlay Network Model by the Integer Linear Programming.In *Proc. of the 10th IEEE Conf. on Emerging Technologies and Factory Automation (ETFA)*, pages 502–510, Sep. 2005.
- P. Jurčík, A. Koubâa, M. Alves, E. Tovar, and Z. Hanzálek.: A Simulation Model for the IEEE 802.15.4 Protocol: Delay/Throughput Evaluation of the GTS Mechanism. In *Proc. of the 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 109–116, Oct. 2007.
- P. Jurčík, R. Severino, A. Koubâa, M. Alves, and E. Tovar.: Real-Time Communications over Cluster-Tree Sensor Networks with Mobile Sink Behaviour. In *Proc. of the 14th IEEE International Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 401–412, Aug. 2008.
- P. Jurčík, R. Severino, A. Koubâa, M. Alves, and E. Tovar.: On the Capacity of Cluster-tree ZigBee Network. In *Proc. of the 4th International Conf. on COGnitive systems with Interactive Sensors (COGIS)*, Nov. 2009.
- P. Jurčík and Z. Hanzálek.: On the interdependence of reliability, energy consumption and timeliness in IEEE 802.15.4/ZigBee cluster-tree Wireless Sensor Networks. In review for *Proc. of the 8th IEEE International Workshop on Factory Communication Systems (WFCS).*