An Approach to Timeliness in Wireless Sensor Network Communications

vom
Fachbereich Elektrotechnik und Informationstechnik
der Technische Universität Kaiserslautern
zur Verleihung des akademischen Grades eines

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von
Ramon Serna Oliver

D 386

Eingereicht am:	20.07.2010
Tag der mündlichen Prüfung:	30.09.2010
Dekan des Fachbereichs:	Prof. Dipl.-Ing. Dr. Gerhard Fohler
Promotionskommission	
Vorsitzender:	Prof. Dr.-Ing. Wolfgang Kunz
Berichterstattende:	Prof. Dipl.-Ing. Dr. Gerhard Fohler
	Prof. Jean-Dominique Decotignie
	Prof. Peter van der Stok

Abstract

Wireless Sensor Networks (WSN) are dynamically-arranged networks typically composed of a large number of arbitrarily-distributed sensor nodes with computing capabilities contributing to –at least– one common application. The main characteristic of these networks is that of being functionally constrained due to a scarce availability of resources and strong dependence on uncontrollable environmental factors. These conditions introduce severe restrictions on the applicability of classic real-time methods aiming at guaranteeing time-bounded communications.

Existing real-time solutions tend to apply concepts that were originally not conceived for sensor networks, idealizing realistic application scenarios and overlooking at important design limitations. This results in a number of misleading practices contributing to approaches of restricted validity in real-world scenarios.

Amending the confrontation between WSNs and real-time objectives starts with a review of the basic fundamentals of existing approaches. In doing so, this thesis presents an alternative approach based on a *generalized timeliness notion* suitable to the particularities of WSNs. The new conceptual notion allows the definition of feasible real-time objectives opening a new scope of possibilities not constrained to idealized systems.

The core of this thesis is based on the definition and application of Quality of Service (QoS) *trade-offs* between timeliness and other significant QoS *metrics*. The analysis of *local* and *global trade-offs* provides a step-by-step methodology identifying the correlations between these quality metrics. This association enables the definition of alternative trade-off configurations (*set points*) influencing the quality performance of the network at selected instants of time.

With the basic grounds established, the above concepts are embedded in a simple routing protocol constituting a proof of concept for the validity of the presented analysis. Extensive evaluations under realistic scenarios are driven on simulation environments as well as real testbeds, validating the consistency of this approach.

Every action has a consequence[1].

This thesis is a consequence.

[1]From Netwon's Third Law of Motion.

Preface

Pusuing a Ph.D. is –generally speaking– quite an enjoyable experience: one gets to travel around the world attending project meetings and conferences in remote locations, meeting people from pretty much everywhere, and all of this while still maintaining a sort of student profile with the first benefits of an academic career. Nontheless, although the *average* feeling is that of being a full-time student with a paycheck at the end of the month, the *variance* in the *distribution* of collected emotions along the way shows more agitation than that induced by the roller coasters of most amusement parks.

These peaks of alternating tranquility, stress, and over-enthusiasm constitute the basic motivation to keep on going, provided that one gets to master the art of modern academic alchemy[2], defeat procrastination, and manage to maintain contact with the external world, yet avoiding the so feared questions: "– so, when are you finishing?" or "– what are you *exactly* doing?".

Well, to anyone wondering what I have been doing for the last years, I am happy to inform you that the answer is among the coming pages. Hint for the lazy ones: there is a summary at the end!

However, before going to the matter, I would like to express my appreciation and gratitude towards those whom made of both my Ph.D. as well as my stay abroad an –even more– enriching experience.

In first place, to my supervisor Gerhard Fohler for giving me the chance of pursuing an academic career with an incredible international atmosphere. I am truly grateful for all the enriching experiences and good moments accumulated during this instructive period of time.

I would also like to thank very specially all my colleagues from the Chair of Real-Time Systems in TU-Kaiserslautern, both for their support in the scientific research as well as for the many times of fun and procrastination [Cham 10]. In particular, my best wishes are for the (ex-)Ph.D. students Joe Jonsson, Raphael Guerra, Alexander Neundorf, Anand Kotra, Rodrigo Coelho, Stefan Schorr, Viet Cuong and Jens Theis. It was a pleasure to meet you guys and I wish you all the best in your coming achievements!

I cannot do less than show my deepest gratitude to our secretary Stephanie Jung as well as to Carmen Vicente-Fess. Not only for dealing with all the administrative bureaucracy that I have generated along this time, but also for their aid and assistance

[2]Modern academic alchemy consists of the obscure art of converting poured caffein beans into any sort of documented research manuscript (name it paper, article, report, or deliverable).

in the complicated task of being a foreigner in a (sometimes) perfectly-organized country as Germany. I hope that I did not get on your nerves too often with my problems and little questions!

Last, but not least, many thanks to Markus Müller; partly for the technical support but mainly for his patience and tremendous effort in trying to understand my German *chit-chat* and still make the experience enjoyable. *Vielen Dank, Markus!*

During my Ph.D. studies, I have had the pleasure of meeting many other researchers and Ph.D. candidates with whom I had a great amount of interesting conversations. Sometimes they were work-related –others rather not–, but it was always fun to learn and share experiences with so many people coming from different countries and cultures. Definitely, it would have been not the same without such a great and enriching international and intercultural experience.

In particular, I would like to express my gratitude to Peter van der Stok, Jean-Dominique Decotignie, Marc Aoun and Jérôme Rousselot, for all the constructive and certainly profitable discussions that we had during –and after– the *Real-Time/WSN workshops*.

Closing this round of gratitude would not be complete without a mention to Ivan Shcherbakov and Prashant Sachdeva with whom I had the chance to collaborate during their studies in TU-Kaiserslautern.

On a more personal side, I would like to thank my family for giving their support during this long process. Particularly, without the support and encouragement of my parents during my studies this journey would never have started.

Words are not enough to prove my love and gratitude to Raluca, my girlfriend and partner of adventures. I just hope that in the future I will be able to do as much as she has done for me during this period of time.

Finally, coming back to the technical matter, I would like to thank *you* for spending some of your time reading –at least part of– this thesis. I hope that you find it interesting!

Ramon
Kaiserslautern, 20.07.2010

The work presented in this thesis has been partially supported by the European Commission under the Framework 6 IST Project *Wirelessly Accessible Sensor Populations* (WASP, IST-2006-034963).

Publications

I have authored or co-authored the following publications:

Conference and refereed workshop papers

- Ramon Serna Oliver, Gerhard Fohler. *Timeliness in Wireless Sensor Networks: Common Misconceptions* In Proceedings of the 9th International Workshop on Real-Time Networks (RTN2010), Brussels, Belgium, July 2010.

- Ramon Serna Oliver, Ivan Shcherbakov, Gerhard Fohler. *An Operating System Abstraction Layer for Portable Applications in Wireless Sensor Networks.* In Proceedings of The 25th ACM Symposium on Applied Computing (SAC 2010), Sierre, Switzerland, March 2010.

- Jerome Rousselot, Jean-Dominique Decotignie, Marc Aoun, Peter van der Stok, Ramon Serna Oliver, Gerhard Fohler. *Accurate Timeliness Simulations for Real-Time Wireless Sensor Networks* In Proceedings of the European Modelling Symposium 2009, Athens, Greece, November 2009.

- Ramon Serna Oliver, Ivan Shcherbakov, Gerhard Fohler. *Poster abstract: An Efficient Operating System Abstraction Layer for Portable Applications in the Domain of Wireless Sensor Networks.* In Proceedings of The 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09), Barkeley, California, USA, November 2009.

- Ramon Serna Oliver, Gerhard Fohler. *Probabilistic estimation of end-to-end path latency in wireless sensor networks.* In Proceedings of the Sixth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS09), Macau SAR, P.R.C, October 2009.

- Anthony Schoofs, Marc Aoun, Peter van der Stok, Julien Catalano, Ramon Serna Oliver, Gerard Fohler. *A Framework for Time-Controlled and Portable WSN Applications.* In Proceedings of the 1st International ICST Conference on Sensor Networks Applications, Experimentation and Logistics (SENSAPPEAL 2009), Athens, Grece, September 2009.

- Ramon Serna Oliver, Gerhard Fohler. *A proposal for a notion of timeliness in wireless sensor networks.* In Proceedings of the 8th International Workshop on Real-Time Networks (RTN09), Dublin, Ireland, June 2009.

- Ramon Serna Oliver, Gerhard Fohler. *Probabilistic routing for wireless sensor networks.* In Proceedings of the Work-in-Progress Session, 29th IEEE Real-Time Systems Symposium (RTSS08), Barcelona, Spain, December 2008.

Technical Reports

- *Estimation of the probability density function of end-to-end delays in Wireless Sensor Networks*, Ramon Serna Oliver, Chair of Real Time Systems. Technische Universität Kaiserslautern, January 2009.

- *System Models in Wireless Sensor Networks*, Phillip Stanley-Marbell, Twan Basten, Jerome Rousselot, Ramon Serna Oliver, Holger Karl, Marc Geilen, Rob Hoes, Gerhard Fohler, Jean-Dominique Decotignie, Eindhoven University of Technology Department of Electrical Engineering Electronic Systems, May 2008.

Contents

List of Figures

List of Tables

Chapter I

Introduction

The core of Real-Time Systems (RTS) develops from the principles of determinism and time predictability. Processor scheduling, Operating Systems (OS), and task synchronization are examples of relevant problems being currently approach in this area of research.

On the other hand, Wireless Sensor Networks (WSN) pursue ad-hoc communications in environments with a significant uncertainty factor. Energy-aware communications and high scalability are two of the most important factors driving the research efforts in this field.

Merging the two –in principle– divergent research areas is not a trivial task requiring significant efforts in understanding the constraints and limitations of each field as well as their flexibility for adaptations. This chapter introduces the basic principles of each area giving a brief overview of the elemental terminology of each.

The chapter is organized as follows:

Section I.1 introduces the basic principles of RTS and related terminology.

Section I.2 follows with a brief introduction to the properties and basic considerations regarding WSN.

Concluding the chapter, section I.3 overviews the outline of the thesis.

I.1 Introduction to Real-Time Systems

Real-Time Systems (RTS) are particular kinds of hardware or software systems that incorporate timeliness within their target operational parameters. Their correctness is evaluated not only with respect to the set of computed outputs but also taking into account the associated instant of delivery. Predefined timeliness constraints introduced at initial specification phases, condition the entire system development cycle. Among others, common ways to express these constraints include the specification of upper-bounds limits (e.g. deadlines) for the completion of each computation unit (e.g. tasks). These bounds, are later on enforced by dedicated run-time mechanisms.

Examples of RTS are present in different domain fields, like the automotive industry, avionics, control systems, and multimedia entertainment systems. The commonalities among these domains come from the fact that the system is not in direct control of the environment. Contrarily, the environment introduces constraints to which the system has to react within predetermined time intervals.

A RTS in which all time constraints must be strictly enforced (e.g. no deadline miss is allowed) is called Hard Real-Time System (HRTS). Common examples of HRTSs are those present in automotive systems involving safety-critical or essential sub-systems (e.g. airbag, ABS, electronic injection). A classic definition of HRTS states that a deadline miss may result in catastrophic consequences for the user (e.g. life-danger or substantial damages) [Buttazzo 05]. However, as the same notion of HRTS has been applied over time in other non-critical areas like consumer electronics this statement is not accurate anymore. As a matter of fact, the common understanding of HRTSs nowadays establishes that a deadline miss may cause a complete system failure, despite the repercussions of its consequences.

On the other hand, systems with a certain allowance to partially violate their time constraints –under specific circumstances– are known as a Soft Real-Time System (SRTS) [Liu 00]. For example, deadlines may be exceeded on the absence of sufficient system resources, although this is likely to produce a perceivable degradation of the overall system quality. This relaxed form of RTSs introduces the problem of properly defining the tolerance of a system towards unfulfilled timeliness constraints (e.g. deadline misses or undelivered results).

I.1.1 Tasks, Scheduling and System Models

The most well-established achievements in RTSs are related to the *scheduling* of *tasks* in single and multi-processor systems. Task scheduling is the process of organizing the execution of a set of tasks in accordance with a number of possible requirements and constraints. A real-time scheduler performs this planning attending to the temporal constraints of each task and ensures that their execution is compatible with the particular policy that it defines.

Examples of real-time scheduling policies are Earliest Deadline First (EDF) and Rate Monotonic (RM), defined in [Liu 73]. Both methods introduce their respective *models* for the system based on a number of necessary assumptions. System models define a list

of properties that must be fulfilled to guarantee the validity of the method. If the models are obeyed, the defined scheduling policy offers a set of analysis tools and algorithms that will make effective the real-time scheduling of the task set.

New algorithms that produce valid real-time schedules under different system models are continuously proposed (e.g. [Guerra 09]), in particular those following the development of new and challenging system architectures (e.g. server clusters, multi-core processors, distributed systems, etc...). Literature regarding this matter is as extensive as detailed, and it is available for further references (e.g. [Buttazzo 05], [Liu 73], [Burns 90]).

I.1.2 Real-time Networks

The clear analogy between processor scheduling and Medium Access Control (MAC) in networking communications accounts for the application of real-time scheduling policies into real-time message scheduling. From an abstract point of view, messages need to be organized in accessing a shared resource –the communication channel–, comparably to tasks being scheduled for execution in the processing unit.

These scheduling principles have been successfully applied in many communication domains, providing valuable analysis tools to perform real-time analysis. The core of these solutions were originated in industrial applications, initially as wired networks but soon adapted to wireless infrastructures. Real-time methods have been introduced practically in most other networking domains and are still a challenging subject of research in many fields (e.g. avionics, automotive industry). Examples of such methods are present in network protocols like CAN [Davis 07], RTP [Benslimane 00], and TTP [Kopetz 94].

Despite the inherent differences between networking and processing domains, the general approach in real-time networking consists of enforcing –in one way or another– predefined deadlines in the transmission of time-constrained messages. Generally speaking, messages are treated as tasks with respect to their timeliness constraints. The analogy works fine at certain levels of abstraction, although some special considerations apply. For instance, the distributed nature of networks along with the usually-non-preemptible transmission of on-going messages. A common misleading practice is to oversee the necessary adaptations of system models targeting processor scheduling when applied to a new domain. This issue is discussed in detail in section III.

I.2 Introduction to Wireless Sensor Networks

Wireless Sensor Networks (WSN) [Karl 06] are particular ad-hoc networks formed by a set of resource-constrained nodes communicating via hop-by-hop message forwarding. The operational parameters of WSNs are of high variability among different application domains. For instance, the network dimensions as well as density of nodes are two parameters with great variability. Others such as the mobility of nodes, number of data collectors (i.e. sinks) and their exposure to ambient phenomenon influence the particular constitution of these networks.

I.2.1 Typical Applications

The application domain of WSNs is broad and in continuos expansion. It ranges from environmental monitoring of large areas, until in-body healthcare control or structural surveillance. Almost any scenario that implies monitoring of ambient variables which can be perceived through sensors qualifies as a target application for WSNs.

The following is just a representative subset of applications that benefit from WSNs:

Health control – Monitoring of human or animal health parameters are common practices in modern medicine. WSNs offer the possibility of remotely observing the evolution of subjects of interest avoiding intrusive monitoring. Moreover, sensor nodes can be programmed to supply a first-level analysis tool discerning outstanding samples which may require the attention of field experts.

Environmental monitoring – Whether it serves biological purposes or provides assistance in disaster relief operations, environmental monitoring is one of the most extended applications of WSNs. The difficulties of structural network deployments in large areas like forests, under-sea or any other wild natural environment attribute exceptional value to the adaptation of affordable, unintrusive and self-configuring sensor networks.

Intelligent buildings – Temperature, lighting, and regulation of energy consumption are only a few of the many applications carried out in modern building. Increasing the living comfort and reducing the energy waste are two main goals of intelligent buildings which benefit from the versatile functionality of WSNs.

Structural surveillance – Solid materials in buildings, roads, bridges and most architectural constructions suffer from structural stress which can develop in deterioration and in worst cases collapses or serious structural damages. Monitoring these constructions in search of cracks or other signs of structural weakness is an arduous job that can be automated and distributed with WSNs. The increase in terms of accuracy and capability of early detection is notably superior and less costly than any other method involving human supervision.

Security – Intrusion detection and surveillance of restricted access premises are possible with WSNs. Motion tracking and battlefield surveillance are other domains of interest in the development of new protocols and network architectures.

I.2.2 Design Principles

The technology and application domains from which WSNs emerged determine most of the characteristics of their essential design principles. Many of these principles are inherited from embedded systems and ad-hoc networks, while others are particular to sensor networks. The following list summarizes some of the most relevant aspects that arise in designing a WSN:

Distribution – The deployment of most sensor networks come with an implicit need of a distributed organization. Centralized control entities are not an option due to their implications in terms of scalability and adaptivity. Deployed nodes need

to discover by themselves the necessary information to establish communication with one or multiple data collector (i.e. sinks). This *self-organization* is the core of hop-by-hop communications in WSNs and introduces design constraints in most other aspects of the network.

Scalability – Scalable solutions capable of maintaining satisfactory performance levels at variable network lengths are essential to WSNs. The coverage of large areas, dynamic environments, or the sudden variability of network density are some of the many inconveniences that networking protocols may have to deal with. Properly designed WSNs must adapt to these situations while keeping reasonable performance levels.

Adaptivity – There is no common environment for WSN applications. However, most application scenarios include uncontrollable factors to which the network has to adapt. Among them, exposure to nature forces, irregular geography, multiple interference sources, or mobile elements. Thus, it is a strong requirement that WSNs are designed with a high degree of adaptivity towards changing environments.

Data centricity – The collection of information from particular nodes is not an essential service in a typical WSN. Instead, the fundamental design of these networks focuses in gathering significant data, wherever the particular source is. This data centric principle changes the way in which particular elements of the network are located and addressed. It is no more an issue of *who* provides the information, but rather of *which* information is available at a time.

Locality – Multi-hop communications are expensive in terms of overhead and must be kept at a minimum. Maintenance services, discovery algorithms and other network services have to exploit the information provided by nearby nodes and rely in their neighborhood to receive sufficient information for their correct functioning.

Scarce resources – As in any other domain of embedded systems, WSNs are constrained with respect to the availability of resources. Network protocols and other application algorithms have to deal with this shortage and exploit to their best the existing resources.

Energy – Among all available resources, the energy source present on the sensor node constraints the utilization of all other components. Whether it comes from batteries or from other kind of energy harvesting systems, the amount of availabile of energy delimits the overall capacity and performance of a node.

Cost – Most deployments are directly constrained by the production cost of individual nodes. The selection of particular technologies or hardware components my be directed by the available budget and the manufacturing cost.

I.2.3 Ad-hoc multi-hop communication

Many design properties in WSNs coincide in principles with those of Mobile Ad-Hoc Networks (MANET). In particular, with respect to the capabilities of self-organization and hop-to-hop communication WSNs can be classified as a particular kind of MANET.

This is plausible by observing the core of a generic network stack in WSNs.

Despite current trends, which tend to increase the number of networking layers, a typical protocol stack for WSNs consists of a minimalistic subset of the OSI model [ISO 10], namely the physical layer, the MAC layer and the networking or routing layer. A brief description of these layers and their main issues with respect to WSNs follows:

Physical layer – Despite not being tight to any specific technology, the vast majority of WSNs make use of Radio Frequency (RF) transceivers. The main function of this layer is to transform bits into signals, which are then transmitted through the physical medium. Due to the large number of hardware technologies providing alternative physical interfaces, it is a rather complex layer which is widely studied in literature [Karl 06].

MAC layer – The MAC layer imposes an organized control in accessing the medium. Transmitting entities must obey the rules introduced by the MAC protocol to gain access to the channel. Pure Time Division Multiple Access (TDMA) and Carrier Sense Multiple Access (CSMA) are the two types of medium access policies that prevail in WSNs. Most MAC protocols are based on one or the other, and their particularities consist of adjusting their behavior to achieve better results for a reference metric, which in most cases, is the average energy consumption.

Networking layer – Routing protocols provide means to extend the one-hop communication level achieved by the MAC layer into end-to-end communication between any two entities in the network. At a glance, routing protocols in WSNs can be classified in two types, depending on the strategy of nodes to effectively communicate in a self-organized manner:

Re-active protocols try to find valid routing paths as a response to events which need to be communicated. Hence, during periods of inactivity the protocol remains in a latent state, which in turn is directly reflected in a low energy consumption. However, these protocols suffer of a larger latency when it comes to the moment of transmitting the first of each series of messages, which cannot be effective until the path is discovered.

Pro-active protocols solve the problem of large latency by periodically maintaining routing paths despite no message needed to be transmitted. The downside is a clearly larger consumption of energy which in some cases results unnecessarily wasted.

I.2.4 Hardware, firmware, and software

Three main aspects describe the characteristics of a WSN platform: the on-board hardware components assembled on each node, the firmware, composed by the operating system and drivers allowing access to these components, and finally, the software program(s) implementing the network application(s).

Hardware

The basic hardware platform in a typical WSN is composed of a processing unit (CPU), some limited amount of memory, a communication interface (i.e. radio transceiver), and one or several sensors. Micro-Controllers (μC) are commonly adopted due to their compact design and flexibility, although field programmable gate arrays (FPGA) as well as other technologies are also being used. The main constrains that restricts the development of hardware platforms are related to energy consumption and physical dimensions.

Firmware

The operating system, or run-time environment, represents the core of the system firmware. It provides an abstraction layer (i.e. functional API) between the application developer and the general hardware platform. In addition, peripheral drivers and other extensions may allow access to particular hardware components or communication interfaces, in some cases implementing a complete or partial networking stack.

Software

The end-user software application is the collection of programs implementing the main functionalities of the sensor network. These programs may be developed on top of the hardware platform as stand-alone applications, or as high level network services embedded into complex frameworks. Typical programming languages for the former, include *C*, *C++*, and *nesC*, while the latter may take advantage of service-oriented and other general description languages such as *XML*.

I.3 Thesis outline

The rest of this thesis is organized in the following chapters:

Chapter II overviews the necessary background to further develop the contents of this thesis. The chapter presents a prospective analysis of the state-of-the-art and exposes the main open issues and points of interest with respect to real-time and WSNs.

Chapter III analyzes the implications of the commonly adopted notion of timeliness and explores an alternative notion better suited for WSNs. The chapter introduces the generalized timeliness notion with a practical example of its applicability, which estimates the end-to-end latency of a sequence of messages in a sensor network.

Chapter IV presents a general analysis of Quality of Service (QoS) in WSNs. The chapter defines a step-by-step analysis of QoS trade-offs and identifies the main elements involved.

Chapter V describes a timeliness aware routing protocol embedding the generalized timeliness notion and the estimation of the end-to-end latency introduced in chapter III.

Chapter VI presents an evaluative analysis of the timeliness aware protocol based on simulations and small scale test-bed experiments. Main performance figures are presented and discussed in this chapter.

Chapter VII concludes the work presented in this thesis and overviews the main achievements and most meaningful results.

Appendix A describes in extension the Operating System Abstraction Layer (OSAL), which is already introduced in section II.2. The appendix goes in depth with details that could not fit in the main body of the thesis.

Appendix B contains further details and additional figures that were not placed in their respective sections. The purpose of this appendix is to provide higher detail in some particular topics of this thesis without incurring in overloading the main sections.

The complete bibliography and glossary of terms are listed in the following sections. In addition, the highlights and main contribution of each individual chapter are summarized in a closing section for convenience and easy location of concepts.

Chapter II

Overview of the State-of-the-Art

Before aiming at prominent achievements in any research direction it is necessary to understand the current situation of well-established results. In doing so, this chapter presents an overview of the state-of-the-art in the area of Wireless Sensor Networks (WSN), exploring a number of representative scientific publications and related sources.

The chapter is structured as follows:

Section II.1 briefly introduces the architecture of sensor nodes and their hardware components. Part of this section is dedicated to identifying the main sources of energy consumption in a typical sensor node.

Section II.2 explores how the hardware components and resources are handled by the Operating System (OS). It presents a general overview of existing OSs and introduces the basics of a conceptual abstraction layer for the OS.

Once the hardware and firmware aspects of the sensor node are covered, section II.3 explores the real-time aspects in the current state-of-the-art. The section is divided in two parts, going respectively in depth into real-time communication protocols (section II.3.1), and real-time frameworks and middleware (section II.3.2). For practical reasons, and to provide a better understanding of the remaining of this thesis, preference is given to real-time and timeliness-aware protocols.

The purpose of section II.4 is to introduce the basic notions regarding Quality of Service (QoS). In particular, it develops the basic definitions and terminology used throughout this thesis, which is necessary to diminish the possibility of confusions occasioned by terms that are often subject to different interpretations.

WSNs constitute an emerging area of research which combines and shares properties with many other fields. However, there are an important number of inherent properties that represent major challenges by themselves and need to be individually approached. For this reason, section II.5 explores the most relevant properties and identifies common misleading assumptions as well as the consequences associated to them.

Finally, the chapter concludes with a summary.

II.1 Architecture of Sensor Nodes

The architecture of a node in a typical Wireless Sensor Network (WSN) does not differ in many aspects from the general architecture of embedded systems. Sensor nodes are small, low-budget devices with a significantly reduced quantity of available resources. This section overviews the main hardware components in a typical sensor node and identifies the most significants sources of energy consumption.

II.1.1 Typical Hardware Components

The particularities of most application domains introduce a number of constraints regarding size, weigh, and physical dimensions, which impose limitations on the amount of components that can be integrated in the hardware platform.

However, the minimum set-up for a sensor node consists of a processing unit, some memory –in any of its possible variants–, a communication interface (e.g. Radio Frequency (RF) transceiver) and one or more sensors. The connexion between these elements depends greatly on the singularities of each system and its design.

Processing Units

A popular choice among embedded systems is to rely on Micro-Controller (μC) to handle the core functionalities of the system. μC concentrate on a single Integrated Circuit (IC) most of the hardware components required in an embedded platform. Namely: CPU, RAM, programmable memory, communication interfaces, and data buses. Additionally, some μC also embed a set of sensors or a number of A/D and D/A converters.

Several families and models of μCs with particular specifications are available *of-the-shelf*. The *Texas Instruments MSP430* [msp 04] is a popular choice among them, partly for having a design explicitly conceived for low-cost embedded systems. This family of μCs is powered by a 16-bit CPU operating in one of six available power-consumption modes. These provides a range of different states in which unneeded components remain in sleep mode, hence achieving a significantly low energy consumption.

Table II.1, extracted from the technical data-sheets summarizes the general specifications of the MSP430 family.

Alternatives to μCs like field programmable gate arrays (FPGA) or application specific integrated circuits (ASIC) seem appealing due to their high performance at moderate levels of energy consumption. However, the larger flexibility of μCs with respect to development and maintenance of applications prevails over the efficiency of specialized solutions. Hybrid approaches, in which static modules are put into an ASIC, while others prone to be updated after deployment run on μCs seem promising [Karl 06]. Further research on this topic may lead to efficient combination of resources showing the true potential of combined architectures.

- Low Supply Voltage Range $1.8V$ to $3.6V$
- Ultralow-Power Consumption:
  - Active Mode: $200\mu A$ at $1MHz$, $2.2V$
  - Standby Mode: $0.7\mu A$
  - Off Mode (RAM Retention): $0.1\mu A$
- Five Power Saving Modes
- Wake-Up From Standby Mode in less than $6\mu s$
- 16-Bit RISC Architecture, $125ns$ Instruction Cycle Time (at 8Mhz)
- Basic Clock Module Configurations:
  - Various Internal Resistors
  - Single External Resistor
  - $32kHz$ Crystal
  - High Frequency Crystal
  - Resonator
  - External Clock Source
- 16-Bit Timer_A With Three Capture/Compare Registers
- On-Chip Comparator for Analog Signal Compare Function or Slope A/D Conversion
- Serial Communication Interface (USART0)
  - Software-Selects Asynchronous UART or Synchronous SPI
- Serial Onboard Programming,
  - No External Programming Voltage Needed
  - Programmable Code Protection by Security Fuse
- Family Members Include:
  - MSP430F122: 4KB + 256B Flash Memory 256B RAM
  - MSP430F123: 8KB + 256B Flash Memory 256B RAM

**source: MSP430 Technical Data Sheet [msp 04]*

Table II.1: *MSP430: Summary of specifications.*

Memory

Memory plays two important roles in any embedded device. On the one hand, it holds the necessary program to govern the system. On the other, it provides the necessary capacity to store and access data. The two roles have different requirements which are often fulfilled by different kinds of memory.

Read Only Memory (ROM) is usually preferred to store executable code in a non-volatile manner. This can be erasable and re-writable (e.g. EPROM, EEPROM, FLASH) or not (e.g. PROM). The amount of ROM in the system determines an upper bound for the size of the program that can be installed on the system at once. The capability of re-programming the memory determines if the application code can be updated during or after deployment.

Random Access Memory (RAM) comes in many different technologies, sizes, and specification. The most important characteristic that identifies RAM is related to the possibility of randomly accessing its data. RAM is typically *volatile*, i.e. data does not persist in memory after the loss of energy. The use of RAM is mostly limited to the storage of data and the maintenance of run-time system status (e.g. program variables and internal data structures).

Communication Interface

Although WSNs are not constrained to the RF spectrum, this is the most commonly used medium in wireless communication. Other options, like optical communications present severe disadvantages which dissuade their application in the field. In contrast, RF communications provide significantly long transmission ranges at high data rates and do not require line of sight between two communicating entities.

As an example, Table II.2, extracted from the technical data-sheets summarizes the general specifications of the popular CC2420 family of transceivers [cc2 08].

Sensors

The main purpose of WSNs involves sensing physical variables. A great variety of sensors, which keeps on growing as technology advances, is available on the market. However, as the main part of this thesis is not affected by the specific sensing capacity of nodes, the analysis of particular sensors is intentionally left out. Nevertheless, the following is a brief summary introducing a number of common sensors and their typical use.

Acoustic sensors – In this category, microphones and ultrasonic sensors are included. Their main purpose is to allow capturing different levels of noise and convert them into digital signals.

Thermal sensors – These include thermometers, thermal flux sensors, and other sorts of temperature measuring devices.

- True single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver with baseband modem and MAC support
  - DSSS baseband modem with 2 MChips/s and 250 kbps effective data rate.
  - Suitable for both RFD and FFD operation
- Low current consumption (RX: 18.8 mA, TX: 17.4 mA)
  - Low supply voltage (2.1 – 3.6 V) with integrated voltage regulator
  - Low supply voltage (1.6 – 2.0 V) with external voltage regulator
- Programmable output power
  - No external RF switch / filter needed
  - I/Q low-IF receiver
  - I/Q direct upconversion transmitter
  - Very few external components
  - 128(RX) + 128(TX) byte data buffering
  - Digital RSSI / LQI support
  - Hardware MAC encryption (AES-128)
  - Battery monitor

**source:CC2420 Technical Data Sheet [cc2 08]*

Table II.2: *CC2420: Summary of specifications.*

Environment sensors – These sensors are used to detect special weather conditions, including humidity sensors, dew warning alarms, moisture detectors and rain sensors.

Optical sensors – This category refers mainly to light sensors, although there are others like color detectors and infrared sensors.

Motion sensors – Several sensors fit in this category, including accelerometers, gyroscopes, inclinometers, and multiple positioning sensors.

Medical sensors – Medical conditions are monitored with special sensors like –among others– blood pressure sensors and ECGs.

II.1.2 Main Sources of Energy Consumption

The overall energy consumption of the hardware platform is a critical aspect to take into consideration during the selection of a particular sensor node. In equal conditions, the particular technology and number of present components accounts for the life-time of a node upon deployment. In general, strong restrictions come from the need of keeping a low energy consumption profile, despite the possibility of adding energy harvesting elements (e.g. solar panels).

This section overviews the main sources of current consumption in a typical WSN.

The examples and indicative figures are taken from official data-sheets as an estimated guideline. However, as technology is rapidly evolving, it is difficult to commit to any statement of validity for these figures beyond an approximate order of magnitude. Certain elements, and in particular some sensors, are left aside of this analysis as they require specific considerations depending on the target application domain.

Micro-Controller

Modern μCs can operate in one of multiple modes, each of them enabling or disabling a subset of features to achieve a certain energy consumption level. Taking as an example the MSP430, figure II.1 –extracted from the MSP430 technical data-sheets– shows the energy consumption figures for each operating mode. Considering the difference between the least and most consuming modes, there is a difference of approximately $350\mu A$.

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
$I_{(AM)}$	Active mode	T_A = –40°C +85°C, $f_{MCLK} = f_{(SMCLK)}$ = 1 MHz, $f_{(ACLK)}$ = 32,768 Hz, Program executes in Flash	V_{CC} = 2.2 V		200	250	µA
			V_{CC} = 3 V		300	350	
		T_A = –40°C +85°C, $f_{(MCLK)} = f_{(SMCLK)} = f_{(ACLK)}$ = 4096 Hz, Program executes in Flash	V_{CC} = 2.2 V		3	5	µA
			V_{CC} = 3 V		11	18	
$I_{(CPUOff)}$	Low-power mode, (LPM0)	T_A = –40°C +85°C, $f_{(MCLK)}$ = 0, $f_{(SMCLK)}$ = 1 MHz, $f_{(ACLK)}$ = 32,768 Hz	V_{CC} = 2.2 V		32	45	µA
			V_{CC} = 3 V		55	70	
$I_{(LPM2)}$	Low-power mode, (LPM2)	T_A = –40°C +85°C, $f_{(MCLK)} = f_{(SMCLK)}$ = 0 MHz, $f_{(ACLK)}$ = 32,768 Hz, SCG0 = 0	V_{CC} = 2.2 V		11	14	µA
			V_{CC} = 3 V		17	22	
$I_{(LPM3)}$	Low-power mode, (LPM3)	T_A = –40°C	V_{CC} = 2.2 V		0.8	1.2	µA
		T_A = 25°C			0.7	1	
		T_A = 85°C			1.6	2.3	
		T_A = –40°C	V_{CC} = 3 V		1.8	2.2	µA
		T_A = 25°C			1.6	1.9	
		T_A = 85°C			2.3	3.4	
$I_{(LPM4)}$	Low-power mode, (LPM4)	T_A = –40°C	V_{CC} = 2.2 V/3 V		0.1	0.5	µA
		T_A = 25°C			0.1	0.5	
		T_A = 85°C			0.8	1.9	

Figure II.1: *Current consumption and description of power modes in the MSP430. Extracted from the MSP430 data-sheets [msp 04].*

RF transceiver

In terms of current consumption, RF communications are among the most expensive operations in WSN. Table II.3 shows the typical current consumption of the CC2420 extracted from the transceiver's data-sheets.

A comparison between this figure and the previous one shows that the current consumption of a CC2420 in both "receive mode" and "transmit mode" is one or two orders of magnitude higher than in the active mode of the MSP430. This is a clear motivation for keeping radio activity at a minimum if energy efficient networks are envisioned. The

role of power-aware protocols in reducing the active time of radio transceivers is essential in saving energy and enlarging the system life-time.

Mode	Consumption
Idle mode	$426\mu A$
Receive mode	$18.8mA$
Transmit mode:	
$-25dBm$	$8.5mA$
$-15dBm$	$9.9mA$
$-10dBm$	$11mA$
$-5dBm$	$14mA$
$0dBm$	$17.4mA$

Table II.3: *Current consumption and description of power modes in the CC2420 [cc2 08].*

Memory

On-chip memory consumption is typically accounted for on the figures corresponding to the current consumption of the μCs. Among the remaining kinds of memory, the most common and with a higher impact on the overall current consumption is FLASH memory.

As an example, in [Mainwaring 02] the authors estimated the power consumption of continuous reading and writing operations on the integrated FLASH memory of a Mica sensor node [Hill 02]. The result shows that while reading data the current consumption is $1.111nAh$, and while writing or erasing the consumption raises up to $83.333nAh$.

One important characteristics of FLASH memory is that erasing is done in blocks (or pages) which sizes varies between models. Note, in any case, that in terms of power consumption writing and erasing operations are nearly one order of magnitude more demanding than reading operations.

Life-time vs Energy consumption

There is a relation between the energy consumption of hardware components and the effective life-time of sensor nodes. Under strict energy requirements, in [Mainwaring 02] the authors detail the energy strategy to achieve a network life-time of 9 months in a network of Mica motes. Based on a pair of AA batteries (approximately $2200mAh$ at $3V$), the authors estimate a conservative daily available current budget of $6.9mAh$. The current consumption is distributed among the node's operations as shown in table II.4.

Table II.5 extracted from [Rossi 10] provides the life-time estimation for three sensor boards. The table is based on a mesh network of nodes powered with a pair of AA batteries reporting data once every 3 minutes.

Operation	nAh
Transmitting a packet	20.000
Receiving a packet	8.000
Radio listening for 1 millisecond	1.250
Operating sensor for 1 sample (analog)	1.080
Operating sensor for 1 sample (digital)	0.347
Reading a sample from the ADC Flash	0.011
Read Data Flash	1.111
Write/Erase Data	83.333

Table II.4: *Power required by various Mica operations (extracted from [Mainwaring 02]).*

Mode	Mica2	MicaZ	Telos
Wake-up delay	$0.2ms$	$0.2ms$	$0.006ms$
Sleep mode	$30\mu W$	$30\mu W$	$2\mu W$
Active mode	$33mW$	$33mW$	$3mW$
Data rate	$19kbps$	$250kbps$	$250kbps$
Life-time	453 days	328 days	945 days

Table II.5: *Example of life-time vs energy consumptions on three sensor platforms.*

II.2 Operating systems

Little work has been done towards the unification of the existing variety of Operating Systems (OS) targeting sensor networks. Well established standards like POSIX [Rivas 03], including specific profiles for embedded systems, are too complex for the limited availability of resources in typical sensor platforms.

The following list is representative of the current state-of-the-art. It provides a general overview of the services offered by a typical OS in WSNs as well as of existing tools to assist programmers in building and testing their applications.

II.2.1 Thread-based Operating Systems

Thread-based operating system have the advantage of following a well-known and extended programming model. They introduce a minimal learning-curve for application programmers with experience in most general purpose OSs. However, in systems with low availability of resources, the overhead produced by the scheduler and the handling of tasks may result significantly large. Nevertheless, there are a number of available thread-based OSs for WSN approaching in different manners the trade-off between offering a full-featured OS and incurring in relatively low overhead.

Mantis OS

Mantis OS [Bhatti 05] aims at a low memory footprint, easy to program architecture, and support for preemptive multiple threads. It follows a prioritized threaded programming

model similar to classic POSIX with a scheduler based on priorities and a round-robin policy for those threads with same priority.

The OS consists of a kernel with integrated scheduler, a command server and device driver system. It supports mutual-exclusion and semaphores. It also integrates a low-level communications stack for serial and radio communication interfaces, including a simple Medium Access Control (MAC) layer and a device abstraction layer that provides uniform access to devices. MANTIS OS is mostly implemented in C and has been ported to a number of platforms.

FreeRTOS

FreeRTOS [RTE 09] falls in the category of a kernel rather than a full OS as generally understood. It supports, among other features, prioritized and preemptive threads, interrupt service routines (ISR), mutexes and queues and dynamic memory allocation.

FreeRTOS favors simplicity and portability over optimization. Nearly all the code is written in C, with only a few assembler functions. The scheduler follows either a prioritized preemptive or cooperative scheduler policy, depending on its configuration. In the former, tasks with the same priority share CPU time in a round robin fashion, while in the latter, context switches only occur if a task is blocked or yields.

OS Abstraction Layer

The definition of proper control mechanisms for the hardware platform into software frameworks arises a number of portability issues. The Operating System Abstraction Layer (OSAL) [Serna Oliver 10c] is designed to address these issues and diminish the conflicts between different software and hardware platforms. In particular, it addresses the discrepancies among different OS with respect to their functional Application Programming Interfaces (API), hardware configuration mechanisms, resource management and peripherals handling.

OSAL is an abstraction layer set on top of the OS, which translates system primitives from the target glsOS into an unified API. Thus, application builders make use of a common API and hence, portability among different platforms is reduced to the re-implementation of the OSAL.

OSAL embraces the management of hardware configurations and access to specific set-points which represent a major hook to performance trade-offs. It defines a subset of OS primitives which satisfies the basic application builder's requirements but at the same time, remain simple to match most target OSs. The resemblance of the subset to a POSIX [Rivas 03] OS API is motivated by the desire of reducing the learning-curve as well as the preference of a neutral reference without specific features from any particular platform.

II.2.2 Component-based Operating Systems

Component-based OSs are particular kinds of event-driven OSs. They achieve a significantly low overhead due to their simplified scheduling mechanism. However, their

programming models may introduce higher learning times for unexperienced application programmers. In some cases, the benefits of their low overhead is affected by the difficulties of creating efficient implementations.

Contiki

Contiki [Dunkels 04] is an operating system designed for memory-constrained environments, such as sensor networks. It is built around an event-driven kernel [Duffy 07], and features dynamic loading and unloading of individual programs and services. It supports a full TCP/IP stack via the μIP library, as well as programming abstractions via protothreads. Contiki is implemented in C and has been designed to be easily portable to new platforms.

TinyOS

TinyOS [Berkeley 09] is a popular run-time system specifically designed for networked sensors. Its wide adoption is one of its main strengths together with the availability of a rich library of networking and application components.

TinyOS implements an event-driven execution paradigm where every execution is triggered by some external event interrupt. It provides physical device abstractions as a conventional OS does. The programming model exposed by NesC incorporates the event-driven execution, the concurrency model, and the component-oriented application design of TinyOS. However, it requires the programmer to adopt a very careful programming policy which may jeopardize the ability of reusing code in various deployments.

II.2.3 Virtual Machines

An alternative approach to achieve the abstraction of OSs, hardware platforms, and architectures is to develop on top of Virtual Machines (VM). Squawk, Maté [Levis 02], Sentilla [Corporation 09] and other VMs are built to target sensor networks and small embedded systems. Despite the benefits of VM in terms of quick deployment and maintainability, it is arguable that the overhead in terms of execution and memory utilization represents a serious inconvenience for complex deployments.

VMs are typically reserved for simple monitoring activities requiring low effort and minimal configurations. This enables unexperienced users to adopt sensor networks without requiring the assistance of experts. However, the performance and flexibility of these systems is not comparable to that of systems running native applications. Nevertheless, further literature on this topic is available for reference in [Costa 07].

II.3 Real-Time Aspects

Adapting conflicting requirements and limitations of Real-Time Systems (RTS) and WSNs and bringing both domains into a harmonic coexistence is a challenging problem that has been approached from different perspectives. Existing methods propose a

number of solutions and achieve –in some cases– well defined real-time methods for WSNs.

Individual communication protocols introduce support for real-time within the typical services provided by their respective networking layer. A bottom-up approach guarantees the interoperability of different protocols as they are combined according to the necessities raised by the particular application. However, such designs are typically dependent on the accuracy and performance of lower and upper layers, and only a careful combination with attention to their specifications may lead to complete real-time networking stack.

Integrated approaches to communication frameworks with real-time support are an alternative to the classic layered stacks. They offer closed solution without a clear separation of layers allowing shortcuts that would not be possible in a layered architecture. However, their flexibility in adapting to new scenarios which were not considered at design time are significantly reduced.

In both cases, the effectivity of these solutions depend in great measure on the characteristics of system models and their particular interpretation of real-time objectives. The following is a retrospective analysis of existing methods representative of both categories.

II.3.1 Real-Time Communication Protocols

The following sections overview the current state-of-the-art in real-time networking protocols for WSN. The elaboration of a complete list of existing protocols is not feasible due to the extensive literature on this field. However, in this section a significant number of representative protocols is described, with especial attention to the timeliness related issues as well as particularities of each method.

Real-Time MAC Protocols

MAC protocols [Demirkol 06] are responsible of the coordination between neighboring nodes in accessing a shared communication channel. Real-time communications demand a deterministic resolution of any medium access conflict in order to achieve time-bounded transmissions. Whether it is possible to achieve strict real-time guarantees or not depends in first instance on the capability of the underlying MAC protocol to grant access to the channel within a delimited time interval[1]. A rough overview of the existing literature [Demirkol 06] allows us to classify real-time MAC protocols into two main categories:

The first category includes all those protocols derived from pure Time Division Multiple Access (TDMA) [Kulkarni 04] as well as synchronous and off-line message scheduling. The commonality among them is that they rely on decisions that are either taken

[1] Although many methods claim to guarantee bounded transmission delays, the fundaments of this thesis are based in the impracticability of such bounds. For the sake of completeness, we overview the existing literature as it is originally presented by the authors. However, in section II.5 we discuss in detail the non feasibility of strictly bounded transmission delays in general WSNs

off-line (e.g. message schedule tables) or arranged at run-time during a preliminary configuration phase (e.g. distributed agreement of transmission slots).

The implicit contention-free mechanism of TDMA suggests that theoretical worst-case transmission latencies should be easy to calculate. Each node has a pre-defined schedule that reflects the next transmission slot at any given time. However, the schedule does not account for the queuing mechanism nor manages temporary congestions which may jeopardize the pre-computed transmission bounds. Moreover, the average latency of such schemes tends to be larger than other solutions as they are designed to optimize energy consumption rather than any timeliness aspect. Nevertheless, several real-time adaptations that partially overcome these and other issues have been presented:

[Caccamo 02] proposes a prioritized protocol aiming at hard real-time in WSNs. The protocol relies on a particular network topology, in which nodes are organized in hexagonal cells. Intra-cell transmissions are scheduled according to Earliest Deadline First (EDF) in a TDMA fashion, while channel allocation for inter-cell communications requires a router placed at the center of each cell. Routers are equipped with two transceivers –one to receive, one to transmit– and assigned one of the 7 available frequencies following a Frequency Division Multiple Access (FDMA) scheme. Needless to say, the singularity of the required topology, cost of the equipment, and strong synchronization dependencies limit in great measure the applicability of this solution.

[Sahoo 07] presents RTMAC, a real-time MAC protocol providing delay guarantees. Nodes are organized in cluster and their time-slot are calculated and assigned by the corresponding cluster head –a special node with additional properties. RTMAC is designed to reduce the high latency of classic TDMA protocols, allowing the re-utilization of slots by non-interfering nodes. The protocol takes advantage of the geographical position of nodes, in particular their relative angle with respect to the cluster head, which is used to determine non-interfering nodes. However, such a degree of precision in the estimation of relative positions is challenging for a typical WSN.

More recently, [Gobriel 08] describes a number of promising adaptations to a classic TDMA scheme specially addressed to WSNs. Parallel transmissions are solved with specific graph-coloring heuristics, which improve the transmission latency while keeping a low energy profile. The authors introduce the concept of *slot stealing*, which defines a priority scheme for nodes to steal slots when the original owners leave them unused. Nevertheless, it is questionable whether the efficiency of these heuristics remains acceptable in real-world scenarios.

TreeMAC, presented in [Song 09], focuses on real-time high-data-rate sensor networks. As other TDMA-based protocols, it divides time into slots and assigns them to individual nodes. However, TreeMAC differentiates in that the number of assigned slots per node is proportional to their hop-distance to the sink. Hence, the number of slots assigned to a node of depth l in the network tree is –at least– equal to the sum of those from all its child nodes (i.e. depth $l + 1$) plus its own requirements. TreeMAC addresses the natural increasing congestion in tree-based schemes as the messages approach the sink (i.e. the closer a node is from the sink the higher is the traffic).

The second group embraces protocols based on asynchronous medium access, similar to those of general purpose networks (e.g. Carrier Sense Multiple Access (CSMA)). They

aim at extending the default fairness-oriented behavior of classic MAC protocols offering real-time performance. Authors often make use of hybrid solutions to achieve *pseudo*-bounded delays with a better average performance than pure-TDMA. Nevertheless, most solutions consist of a CSMA scheme combined with other methods like TDMA, prioritization of messages, queuing policies, or specific network architectures.

[Watteyne 05] and [Watteyne 06] propose a hybrid MAC protocol for low-cost hard real-time WSN. Their solution considers a linear network with all nodes separated at least a distance d_{min}. Nodes in position to transmit wait for a back-off time proportional to the distance between them and the last transmitting node. Hence, the further node is granted with the channel access. In addition, two functional modes are possible: protected and unprotected. Upon collision, the network switches to a cell-organized protected mode which guarantees collision-free transmission based on a channel reservation mechanism. By alternating between modes, the protocol is able to provide real-time support while still keeping acceptable average performance. Nevertheless, the topology requirements and synchronization constraints restrain the applicability of this solution.

Z-MAC, presented in [Rhee 08] tries to combine the strength of TDMA and CSMA. It reacts to the level of contention by dynamically switching between the two schemes. As many other solutions, the aim of Z-MAC is to reduce the transmission delay while achieving a good average performance. There are few solutions that explicitly pursue real-time performance; among them, schemes like [Kumar 09] claim to guarantee bounded delivery delays with pure CSMA. However, the constraints imposed by the necessary models or the particularities of the evaluation scenarios makes them impractical for general deployments.

Real-Time Routing Protocols

Routing protocols for WSNs [Al-Karaki 04] present a significant number of challenges in comparison with those of general communication networks. Despite the already mentioned resource constraints, these protocols have to deal with inherent properties of sensor networks. Among them, a strong requirement for self-organization of nodes along typically large networks, as well as unbalanced communication load due to the asymmetry of message flows which converge to one or a few data collectors.

SPEED [He 03], [He 05] is one of the most popular reference papers within the real-time community. This routing protocol introduces traffic control mechanisms to guarantee a desired forwarding speed across the multi-hop network. Speed obeys a stateless non-deterministic geographic forwarding scheme. The forwarding node chooses among its neighbors one of those capable to provide a satisfactory ratio between the forwarding latency to the next hop and the achievable increment in distance (i.e. speed). However, if none of the neighbors is able to satisfy the required speed, the packet is dropped with a certain probability. Moreover, congested areas are avoided with a back-pressure mechanism. The main drawback of SPEED is that its real-time guarantee mechanism is based on traffic control and message flow balancing. Messages that cannot be delivered with their required end-to-end delay are systematically dropped to reduce the network load. Therefore, guarantees are not effectively enforced and the performance is highly dependent on the network load.

MMSPEED [Felemban 05],[Felemban 06] is an extension to SPEED that includes traffic differentiation. For each kind of traffic, it provides multiple packet delivery speeds. Similarly to SPEED, MMSPEED allows message dropping if a certain threshold is reached before a successful delivery is achieved. In addition, it supports probabilistic end-to-end guarantees by introducing probabilistic multi-path-forwarding. Hence, messages are forwarded by multiple paths until a desired probability of delivery success is reached.

Note that both SPEED and MMSPEED are based on geographic forwarding, which assumes that nodes are location-aware. Equipping nodes with the required resources to obtain precise position coordinates (e.g. GPS devices) is often above budget, and the effects of imprecise low-cost solution may jeopardize the protocol performances.

RPAR [Chipara 06], tries to approach end-to-end delay guarantees with the additional consideration of maintaining a low energy consumption profile. It is based on a power-aware forwarding policy with the assumptions of static network and location-aware nodes. The forwarding mechanism tries to adjust the selected transmission-power at the radio transceiver to achieve the lowest possible energy consumption while maintaining the delay guarantees.

In [Soyturk 08], the authors introduce a similar approach named SWR making use of multi-path message forwarding. In SWR, each node has a weight, which can be determined according to multiple parameters. When a message is sent, the weight of the sender is annotated in a specific header field. Transmissions are done by broadcasting the message. Further, each successive hop retransmits the message if the value of its weight field is within the node's own weight and that of the sink. This controlled flooding mechanism generates multiple paths to the sink, while the use of additional Quality of Service (QoS) parameters allows SWR to reduce the energy consumption while providing a satisfactory end-to-end delivery ratio.

In [Shashi Prabh 07], the authors explore conflict-free message scheduling technics to provide end-to-end guarantees. Their method introduces no overhead in the construction of a collision-free schedule, which guarantees that messages are successfully delivered within deterministic bounds. However, this method is constrained to hexagonal topologies which may rarely appear in real-world deployments.

The selection of forwarding nodes to define a routing path have been analyzed from different perspectives. In [De Couto 03], the authors perform a series of experiments which conclude that focusing on obtaining the shorter paths is not necessarily the optimal routing strategy. The main reason of their findings lays on the radio anomalies which are discussed in section II.5.

Following a similar approach, [Korkmaz 03] presents a probabilistic analysis with multi-objective optimization criteria to obtain the most bandwidth and delay constrained paths. Adopting a probabilistic approach in which the available bandwidth and delay are characterized by random variables the authors offer a method to compute routing paths satisfying pre-defined QoS objectives.

Despite not presenting a particular routing strategy, it is worth to mention the work presented in [Wang 09]. The authors provide a complete analysis of the latency of an end-to-end transmission in a WSN which may be valuable in the construction of

timeliness methods. The strength of this method lies on the decomposition of delays and the statistical characterization of the end-to-end latency of messages.

A similar method approached from a different perspective is described in [Serna Oliver 09b]. Chapter III presents an extended discussion on the details of this approach.

II.3.2 Real-Time Frameworks and Middleware

Middleware and frameworks provide a set of services allowing the interaction of nodes with the guarantee of particular properties. The distinction between middleware and frameworks is, to some extent, arbitrary. Frameworks tend to specify a set protocols or a cross-layer approach providing these services. On the hand, middleware may be stand alone layers providing additional services on top of the basic network stack. However, different authors refer to their approaches as one or the other without a clear distinction.

The definition of general frameworks and middleware for WSNs has been explored by different authors. In [Hadim 06] and [Henricksen 06] the authors explore in detail the available literature. In most cases, general frameworks and middleware are focused in providing energy-aware solutions with best-effort performance. However, in this section, we overview a number of them providing explicit support for real-time or QoS methods.

RAP [Lu 02] is a real-time communication architecture following a cross-layer design similar to the concepts presented in SPEED. It integrates a query/event service layer; a location-addressed service; a geographic forwarding protocol supported by a velocity monotonic scheduler. At the bottom of the communication stack, a prioritized MAC protocol guarantees the timely transmission of messages.

The authors of RAP present an extension in [Abdelzaher 03] including a transport protocol which allows a higher degree of abstraction of the communication process. The same work introduces a programming language that completes the framework providing abstract representation of the physical elements conforming the network.

The work in [Yu 03] explores the challenges and issues in designing middleware for WSNs. It presents a cluster-based middleware architecture which provides a VM model for diverse applications on WSNs. The work considers QoS requirements in its design, although it is unclear which mechanisms enforce them.

In [Sharifi 06], the authors present a middleware mechanism explicitly supporting QoS. The proposed cluster-based organization follows a publish-subscribe model taking advantage of the redundancy of service providers to increase fault tolerance. The method is evaluated with respect to real-time support as well as energy consumption.

More recently, the work described in [de Freitas 08] offers a real-time perspective of a flexible middleware aiming at providing QoS support in heterogeneous sensor networks. The authors make use of aspect and component oriented models in combination with a mobile multi-agents approach. Finally, the work introduces a control protocol to reduce the unnecessary message exchange.

II.4 Principles of Quality of Service

The evaluation of complex systems requires the establishment of measurable objectives and proper evaluation criteria. These elements are often referred as QoS. However, the definitions of QoS and other related terms are not homogeneous among existing literature work.

This section tries to bring up the essential terminology and establishes a solid meaning for terms with divergent interpretations. QoS is often referred to as an intrinsic property somehow linked to the system performance, but rarely expressed as a well-defined set of quantifiable metrics. The following is a list of common terms and particular definitions adopted in the reminder of this thesis.

QoS Terminology:

QoS Metrics – A QoS metric (or just metric) is a measurable evaluation criterion reflecting a quantifiable property of the observed system. Implicitly, the definition of a metric may determine the criteria and range of values which express different quality levels. Examples of metrics are: "message latency", "energy consumption per message".

QoS Requirement – A QoS requirement is a constraint set on a QoS metric used to evaluate the system performance. Generally, QoS requirements are set to measure the system performance, although this can be done by multiple metrics. Examples of requirements are: "all messages arriving within $10ms$", "energy consumption per message bellow $100\mu J$".

QoS Level – The level at which a QoS requirement is satisfied is called the QoS level. Some requirements may allow multiple degrees of fulfillment while others can only be fully fulfilled or else completely unfulfilled. A QoS level may be indicated by a percentage of fulfillment or as a binary value.

QoS Parameters – Parameters are system variables that determine the behavior of the system itself. The value of a given parameter can be observed, but not always configured. The nature of parameters and their impact on system properties determine whether these can be influenced by the system user (controllable parameters) or they are governed by uncontrollable sources (uncontrollable parameters). Controllable parameters include system configuration points (e.g. protocol parameters, system set-up). On the other hand, uncontrollable parameters refer to those given by the environment or fixed by design (e.g. ambient temperature, radio propagation speed, hardware design).

QoS Set-Points – Controllable parameters may be adjusted independently or present correlations in their set of possible values. The interaction between conflicting parameters may lead to unfeasible set of values which cannot be applied to the system. A set-point is a set of feasible parameter values that satisfy the QoS requirements at a QoS level. Optimal set-points are those which fully satisfy all QoS requirements.

QoS Trade-Offs – In some cases, multiple requirements can only be satisfied with unfeasible combinations of parameter values. Hence, only sub-optimal set-points are possible, non of them leading to the complete fulfillment of all requirements. A QoS trade-off is a compromise among all the feasible set-points that satisfies a maximization criteria for the QoS level of each QoS metric.

II.5 General Misconceptions

Existing timeliness solutions aim at enabling WSNs to operate with real-time guarantees. However, the design and evaluation of these methods are often based on naive assumptions that constrain their applicability in real-world deployments.

The definition of ambitious goals –which cannot be satisfied unless severe assumptions are granted– is one of the major drawbacks of current real-time methods. This overestimation of capacity entails important simplifications during the evaluation process. Among others, common practices include the definition of misleading evaluative criteria and the loss of generality due to ad-hoc test-beds. Hence, the quality assessment from a timeliness perspective becomes unclear because the methods are evaluated against unrealistic models.

In some cases, general hardware platforms are extended providing additional functionalities based on theoretical requirements to enable better QoS methods. However, some of these solutions overlook at the additional overhead and cost implications of these components. The consequences of unrealistic hardware properties as well as excessive resource demands affect the validity of general real-time and QoS methods.

This section, presents –from a timeliness perspective– three main aspects of existing real-time solutions for WSNs. First, the section overviews the different goals of existing real-time methods deeming their suitability if applied to a realistic WSN. Secondly, it evaluates the consequences of a number of implicit and explicit assumptions taken in the design of these methods. In most cases, these assumptions have a significant impact on the real-time performance and constrain the applicability in real-world deployments. Lastly, the section examines different evaluation criteria and identify common misconceptions of the evaluative process that can lead to misguided conclusions.

The analysis of real-time methods carried out in this section is based on typical software and hardware platforms for WSNs. In particular, the analysis of misconceptions contemplates the consequences of explicit and implicit assumptions regarding hardware properties or unconventional resources. Based on this analysis, the section concludes with a number elementary considerations, which allow mitigating the impact of unrealistic assumptions and facilitate meaningful evaluation tests that increase the confidence of these methods.

II.5.1 Misleading Real-Time Objectives

The solid background of real-time systems is in many aspects a source of inspiration to provide real-time support in new research areas. However, the inherent capacity of satisfying real-time constraints may be significantly different from one domain to

another. Overseeing the fundamental incompatibilities between both domains represents one of the most common misconceptions in real-time methods for WSNs.

Assumption 1 *The goal of a real-time method is to provide hard real-time guarantees for each transmitted message.*

The notion of hard real-time systems [Buttazzo 05], in which each event is associated with a strict deadline, does not match with the general architecture of WSNs. Messages are transmitted via hop-by-hop forwarding through unreliable links; low end-to-end delivery ratio may occur [Zhou 04]; and the low-energy profile of most communication stacks increases the probability of expiring the maximum retransmission attempts without success. A consequence of these facts is that any individual transmission is susceptible to fail.

Guaranteeing strict deadlines requires excessive resources and complex algorithms for which WSNs are not designed. A more elaborated notion of timeliness and the definition of adequate metrics to evaluate the QoS, accommodate to a larger extent the inherent properties of WSNs. For example, in [Serna Oliver 09c] the authors present a notion of timeliness, which based on the current real-time performance of the network, extracts the probability of messages being transmitted within bounded intervals. This approach is explored in detail in III.

Efficient real-time methods should encourage the analysis and exploitation of network trade-offs, adapting their timeliness performance to the suitability of expending resources.

II.5.2 Common Protocols Assumptions

Existing protocols are not free of assumptions. In this section we enumerate a number of misleading assumptions in existing protocols and their implications in realistic scenarios.

Assumption 2 *Availability of resources*

In a number of existing protocols, it is common practice to base the methods on the assumption of specific hardware resources. Although it is possible to conceive a plausible scenario to justify these assumptions, they are not valid for the general case as it delimits the applicability of these solutions to ad-hoc platforms. For example, GPS devices are mentioned by [He 05], [Chipara 06], and [Felemban 06]; [Caccamo 02] assumes multiple radio transceivers; and [Ergen 06] and [Pothuri 06] provide solution based on unconstrained nodes acting as access points.

Assumptions on such equipment imply the loss of generality and restrain the applicability of these methods to particular cases. Mitigating the implications of such assumptions by alternative methods strengthens both the validity and applicability of the method. However, the consequences of such substitutions may introduce inaccuracies with respect to the dedicated hardware that must be taken into account (e.g. increased overhead or lose of accuracy with respect to dedicated hardware).

Data Link Level

Precise models of radio transceivers and wave propagation through the air are inherently complex due to the interaction of a considerably large number of physical laws. However, their accuracy may determine the validity of real-time models built on top of them. The trade-offs between accuracy and simplicity are not straight forward, and lead to different levels of precision. The following aspects have a significant impact on the data link models.

Assumption 3 *Radio links are symmetric and stable over time. Transmission range follows a radial pattern equal to the interference range.*

This set of assumptions has been widely discussed and refuted. Radio transmissions are neither symmetric nor stable over time as shown in [Woo 03], [Zhou 04]. Both studies conclude that the transmission range of omnidirectional antennas is not regular for all directions and varies over time even in static set-ups. In [Holland 06], the authors experiment with the vertical placement of nodes and conclude that nodes placed a distance above the ground achieve a significant larger transmission and reception range.

From a timeliness perspective, the implications of unrealistic radio models introduce a number of important drawbacks. In the first place, in real-world scenarios the delivery ratio drops due to radio anomalies [Turau 06]. Hence, the necessary mechanisms to ensure successful transmission within strict deadlines must be reenforced. Moreover, further nodes are typically preferred by message forwarders, as they offer a shorter hop distance until the sink. However, these nodes may be located within the boundaries of the effective transmission range, where links suffer from a high bit error rate (BER). In [De Couto 03], the authors explore the use of different metrics other than the *distance-to-sink* in order to determine the quality of paths. Their study reveals that the elaboration of a path metric is not straightforward and may require the combination of different indicators.

Broadcast messages, which are often used to build network trees, also suffer from similar effects. For example, a node closer to the sink will broadcast "HELLO" messages to its neighboring nodes, which will then register the source as the forwarding preference for their traffic. However, some of these child nodes may not be able to send their messages back, either due to the non-symmetric range of the radio devices or because of temporal instability. In [Herms 06], the authors explore further this effect and propose a simple method to determine stable links based on the consecutive reception of enumerated broadcast messages.

Assumption 4 *A radio transceiver is either in transmitting or receiving mode, or turned off.*

The common assumption with respect to the radio transceiver is that at any time, it is either turned off or in one of two possible states: receiving (Rx) or transmitting (Tx). However, the transition between these two modes produces a third state in which the transceiver is neither listening nor sending out any signal. This, in general, is widely neglected in simulation models, despite accounting for a large number of collisions. In real-world scenarios, it introduces a large enough interval

of time –e.g. $192\mu s$ in a TI CC2420 [cc2 08]– between sensing the channel and being able to start transmitting. During this gap of time, other nodes sensing the medium may also start transmitting, which may lead to collisions if both nodes are within their interference ranges.

From a timeliness perspective, the most relevant impact of this effect is again a notable decrease of the effective delivery ratio, which indirectly affects the performance figures of real-time protocols validated against simplistic models.

Assumption 5 *The received signal strength (RSSI) is proportional to the distance between sender and receiver.*

The relation between RSSI and the distance between the communicating parts is not as straight forward as often assumed. In [Newport 07], the authors analyze the signal strength measured at increasing distances and conclude that although *the average* signal strength shows a correlated trend with respect to the distance, this cannot be extrapolated to individual measurements. This conclusion is shared in [Zhao 03], which additionally explores the correlation between signal strength and packet loss. They found out that typically, high signal strength produces low packet loss, although surprisingly, the opposite statement does not necessarily hold.

MAC Protocols

Real-time MAC protocols try to guarantee bounded transmission delays between neighbor hops. Their success depends in great measure on carefully defining the operational boundaries with respect to timeliness. Certain assumptions, as the following, may lead to unsatisfactory results.

Assumption 6 *If no other node in the network is trying to access the medium, the medium is free.*

The assumption of complete isolation with respect the the wireless medium is not safe. Some existing methods (e.g. [Mizanian 09], [Abdelzaher 04], [Watteyne 05]) and most TDMA scheduling policies (e.g. [Sahoo 07], [Mavromoustakis 09]) are designed under the assumption of having a constant amount of network capacity at their disposal.

Nevertheless, communications may still suffer from external interferences and reduced connectivity due to weak link. As a consequence, messages may result corrupted or not transmitted, despite theoretical guarantee of conflict-free communications provided by the protocol.

Protocol designers must take into account that RF communications are prone to uncontrollable interferences that may enter in conflict with TDMA schedules as well as with contention-free intervals. The assumption of a completely isolated environment could be a valid claim for testing purposes. However, the calculation of real-time delay bounds based on this principle is not accurate.

Assumption 7 *WSNs can be organized in fixed topologies which remain stable for the entire network life-time.*

The restriction to a particular network topology is common in some real-time protocols (e.g. [Shashi Prabh 07]). Although it is a legitimate requisite for characteristic scenarios, the implications of such assumptions are questionable in real deployments. In fact, provided that factors such as the radio anomalies discussed in assumption 3 are taken into account, the relation between the physical placement of nodes and their connectivity over time with neighbor nodes is not constant.

Routing Protocols

Routing protocols are not exempt of misleading assumptions which cannot be always taken for granted.

Assumption 8 *Location-awareness.*
Equipping each sensor node with a GPS device is out of budget for most WSN deployments. Realistic assumptions should be made also with respect to the availability of resources. Nevertheless, multiple location algorithms are available and can be combined with real-time methods. However, it is important to consider the unavoidable error of these algorithms in finding the exact position of a node. For example, the performance of routing protocols based on geographic forwarding (e.g. [He 03]) may be directly affected or seriously jeopardized if these errors occur.

Assumption 9 *The maximum length of any routing path is bounded. Hop distance is proportional to physical distance.*
Assuming upper bounds on the number of hops necessary to reach the sink from a given source node (e.g. [Ergen 07]) is a very practical but unrealistic restriction. The elaboration of routing protocols that define the trajectory of messages towards the sink following the "shortest path" may result in low throughput. In [De Couto 03], the authors analyze this effect and provide a number of alternative metrics.

Establishing a realistic upper bound requires strong assumptions on the network dynamics which are often out of control. Nevertheless, the establishment of a bound for the "longest possible path" introduces an implicit constrain in the protocol scalability.

The assumption of correlation between the physical distances and the number of hops in a path is on itself based on the assumption of a uniform distribution of nodes. However, given the arbitrary placement of nodes in typical WSN deployments this cannot be generalized. The distribution and density of nodes may be responsible of non-intuitive relations between the hop count and the physical distance.

Assumption 10 *Messages that are not expected to satisfy their deadlines can be dropped.*
This case may not be considered an assumption but rather a common behavior of real-time routing protocols as a consequence of aiming at strict deadlines (see assumption 1). In most WSN scenarios with timeliness requirements, there is an added value to the *freshness* of data. Following this principle, old messages are

often discarded at intermediate hops if the algorithm estimates that their end-to-end deadline cannot be fulfilled.

However, guaranteeing end-to-end delays is not effective if the protocol itself contemplates the possibility of dropping unsuccessful messages based on estimates. In some cases, receiving old data may produce better results than receiving no data at all. Alternative approaches may consider adaptive methods with the ability of defining flexible deadlines.

II.5.3 Imprecise Evaluation Criteria

Choosing meaningful evaluation criteria has a great impact on the performance figures and the quality of the evaluation procedure. In this section, we discuss some important misconceptions affecting the generalization of evaluation analysis in realistic scenarios.

Misleading Theoretical Proofs

Assumption 11 *Everything can be turned into an analytical expression.*

With the use of properly validated models, meaningful bounds for the network latency or other performance metrics can be inferred. However, in many cases the necessary level of abstraction introduces serious simplifications of complex systems; for example: assumptions about traffic pattern distributions, service times, or the minimum network density.

Analysis of *average-case scenarios* provide theoretical bounds for the figures of interest. However, introducing all possible factors that could interfere in the *worst-case scenario* is practically unfeasible in analytical expressions.

Assumption 12 *The distribution of average (service time/transmission latency/queue size) is constant during the entire network life-time.*

This assumption is correlated with the previous one and reflects the unfeasibility of analytical expressions to capture the dynamic behavior of a WSN.

Simplistic Simulations Models

Assumption 13 *"Our model reflects accurately the physical properties of ...".*

Due to the complexities of physic laws and the propagation of waves, a realistic radio model including all possible anomalies is practicably unfeasible. Channel access, environment, and interferences are as important to model as the method being evaluated. Simplistic models may hide design flaws or applicability limitations that appear in real-world deployments.

Experiments such as [Zhao 03], [Turau 06], and [Yarvis 02] show that the deviation between simulation results and real test-beds are not negligible. However, the additional level of complexity involving a real test-bed is not always affordable.

Nevertheless, an appropriate validation process can lead to sufficient levels of accuracy for the most significant figures. For example, in [Rousselot 09], the authors

profile the necessary steps to achieve accurate evaluations of timeliness protocols with properly tuned simulations.

II.5.4 Considerations

Designing and implementing timeliness methods for WSNs without relying on misleading assumptions is a challenge that still needs further attention. The definition of appropriate objectives and a careful validation of models are crucial to achieve high quality methods.

The following list of considerations summarize the main problems of existing methods, and may help overcome a number of popular misconceptions constraining the quality of timeliness solutions:

- Hard real-time solutions require strict deterministic models that are not compatible with WSNs. Adaptive methods and a proper definition of QoS may reduce the number of necessary restrictive assumptions.
- Realistic radio models are difficult to achieve, yet crucial in the evaluation of timeliness models. The careful validation of data link models plays a significant role in the elaboration of satisfactory methods.
- RF communications in WSNs are typically exposed to many sources of interferences. MAC protocols have to be robust enough to deal with unstable channels and weak links.
- Effective routing protocols should be able to exploit timeliness without requiring restrictive resources. Scalability and adaptiveness are also important figures to evaluate.
- Validation criteria must be consistent with the scenarios for which the evaluated methods are designed. Simplistic models may lead to optimistic figures that do not match the real performance.

II.6 Chapter summary

This chapter overviewed the necessary background to further developing the contents of this thesis. Through its sections, it presented a prospective analysis of the state-of-the-art exposing the main open issues and points of interest with respect to real-time and WSN.

The chapter covered the architecture of typical WSNs including aspects from the software and hardware platform as well as the OS. Further, it surveyed the main aspects regarding real-time and the existing state-of-the-art with respect to the networking protocols. Following, the chapter brought up the essential terminology regarding QoS and establishing a solid meaning for terms with divergent interpretations.

The chapter concluded with an enumeration of common misconceptions and misleading practices in current real-time methods for WSN.

Chapter III

Timeliness in Wireless Sensor Networks

The inherent properties of Wireless Sensor Networks (WSN) introduced in chapter II constitute an unfavorable environment for the enforcement of real-time constraints. The unfeasibility of a typical network stack to guarantee time-bounded point-to-point transmissions clashes with the aimed ambition of fulfilling strict end-to-end delays.

Fundamental aspects suggest that alternative approaches need to be explored if meaningful timeliness guarantees are pursued. These approaches may require a significant change of objectives as well as the establishment of an appropriate set of assumptions suitable for typical WSNs.

This chapter provides a thorough analysis of the essential flaws in the timeliness approach currently exploited in WSNs. Moreover, it explores in detail a new direction towards approaching timeliness guarantees in accordance with the imposed properties and constraints of a typical WSN. The chapter is organized as follows:

Before entering in details, section III.1 settles the terminology and basic definitions which will be used along the rest of the thesis.

Section III.2 is divided in two parts. First, it analyzes the most extended classic notion of timeliness in the area of WSN identifying its limitations and fundamental flaws. Next, it presents a generalized timeliness notion, which provides a means to express meaningful timeliness in typical WSNs without restraining their applicability. The generalized notion differs from the classic in terms of requirements and assumptions, and opens an alternative path to pursue timeliness guarantees.

As a proof of concept, section III.3 presents a probabilistic algorithm for the estimation of end-to-end latency of message transmissions in WSNs. This method is based on the generalized timeliness notion and allows to estimate -with satisfactory accuracy- the distribution function of end-to-end routing paths.

Finally, the chapter concludes with a summary.

III.1 Definition of Timeliness

In general terms, timeliness refers to the occurrence of events at suitable or opportune instants of time. More particularly, in Real-Time Systems (RTS) timeliness is a quality metric that evaluates how opportune is the time of occurrence with respect to the associated event. In other words, the metric evaluates the punctuality (or not) of the event.

As in any other metric, the evaluation of timeliness involves the definition of a measurable requirement that can be quantified. The simplest way to measure a metric is to define a boolean condition that can be either satisfied or not. However, it is also possible to define complex expressions providing a range of values according to different levels of fulfillment.

We refer to the notion of timeliness as the conceptual notion that defines the evaluation of time constraints in a system. The notion itself does not determine the real-time performance of a system, however, it defines the way in which timeliness requirements are expressed and measured. Consequently, it conditions the applicability of real-time methods relying on it and delimits the chances to exploit timeliness.

Examples:

(a) The most extended notion of timeliness in RTSs (e.g. [Buttazzo 05]) allows individual deadlines to be associated for each event (e.g. task instance). Hence, the metric is evaluated as a boolean variable that indicates whether an event occurred within a given time interval or not.

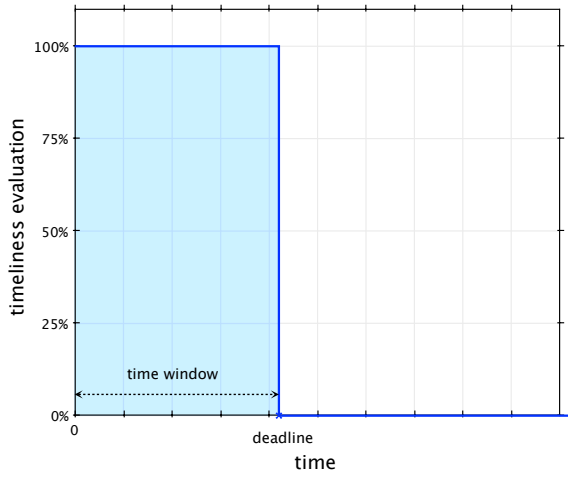
(b) In other systems, the notion defines both an earliest and latest instant of time for the event to occur. As an application example, the time window in which the airbag of a car must be inflated after a collision is tightly determined by physic laws. Either a too early or a too late inflation of the air cushion may result unsuccessful, or even produce further injuries to the passengers. Hence, the notion of timeliness of such a system must allow expressing additional constraints that define bounds for both ends of the interval.

(c) The notion of of (m, k)-firm deadlines [Hamdaoui 95] addresses the characterization of timing constraints of real-time streams. For a stream of events, it evaluates timeliness satisfactorily if at least m out of any k consecutive streams meet their deadlines. Otherwise, it is considered a dynamic failure.

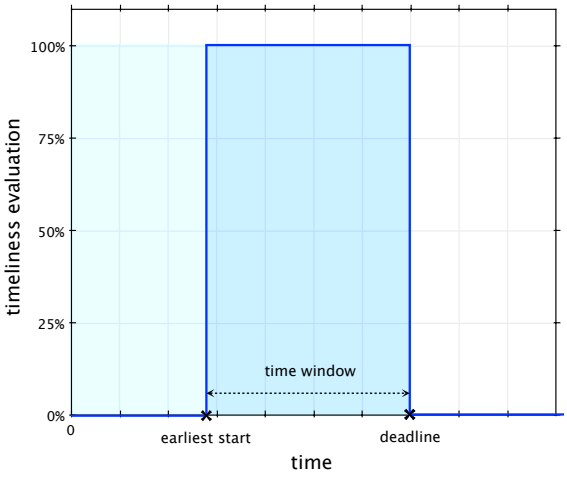
(d) Other notions of timeliness may allow to express a function (continuous or discrete) to evaluate numerically the instant in time when the event occurs (e.g. utility functions [Li 04]). The value is often inversely proportional to the distance between the occurrence of the event and some earlier reference in time. Applications with such requirements include multimedia systems in which media units (e.g. video frames) should be presented to the user at precise instants. Late or early deliveries reduce the perceived quality although to some extent, they may be preferable than no delivery at all.

Figure III.1 illustrates the effects of the above mentioned notions of timeliness on the

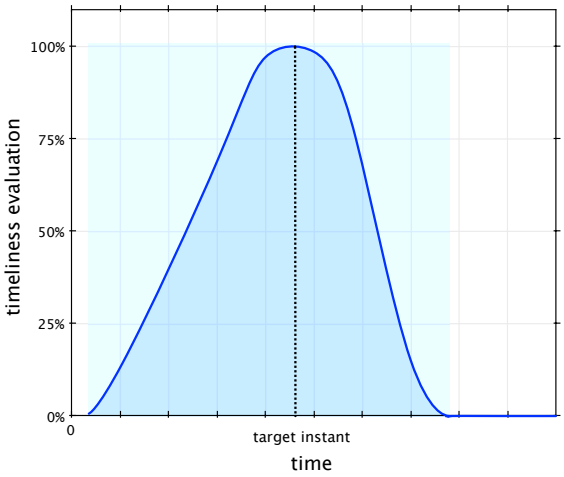
evaluation of the quality metric.



(a) *Expressing timeliness constraints by means of a strict deadline*



(b) *Expressing timeliness constraints by means of earliest release time and deadline*



(c) *Expressing timelines constraints by means of utility function*

Figure III.1: *Effect of the timeliness notion on the evaluation of the quality metric.*

III.1.1 End-to-End Timeliness

The networking nature of Wireless Sensor Networks (WSN) introduces different levels at which timeliness constrains can be expressed and observed. Implicitly, each network layer will have a different view of timeliness. While a MAC protocol will care about the latency to forward a message to an immediate neighbor, routing protocols will deal with a multi-hop scenario with additional impediments not present in the former (e.g. path discovery, cul-de-sacs, etc). Similarly, at the application level, timeliness constraints

are given for functional units (e.g. "report alarm to sink"), which do not reflect the particular vision of underlaying layers.

For higher networking layers (i.e. routing or above), it is common to express timeliness from an end-to-end point of view. The significance of this metric is centered on the total communication latency, measured from the instant of a message being transmitted by the source node until it reaches its final destination. However, possible incidences arising during this process are not relevant to the metric (e.g. the particular path across the network or the number of retransmission required).

This thesis relies on the impossibility of achieving strict end-to-end delay guarantees in WSNs (see section II.5 for argumentation). Nonetheless, the objective of this chapter is to show that it is possible to elaborate a meaningful end-to-end timeliness metric suitable for WSNs.

The following section presents a thorough analysis of the weakness of the classic notion of timeliness and presents a new notion better suited for WSNs.

III.1.2 Timeliness as a Qualitative Metric in WSN

Although timeliness is an important Quality of Service (QoS) metric in most distributed applications, it is often pushed into a secondary role in WSNs. The reason lies in the supremacy of energy consumption as the top priority QoS metric.

Nevertheless, timeliness is an intrinsic quality metric which grows in relevance with the expansion of WSNs to time-sensitive scenarios. The vast majority of applications in sensor network deployments perform monitoring and sensing tasks involving time-critical environmental variables. For example, typical applications include the detection of floods or fire, security surveillance, elderly care, or health monitoring of herds.

In general, the selection of a suitable timeliness notion for a specific system should be directed by its particular constraints. As an example, most Hard Real-Time Systems (RTS) express their time constraints in the way of strict deadlines. This is appropriate when the time constraints are directly extracted from physical properties that force events to occur in a timely manner (e.g. the ABS sub-system in a vehicle is constrained by physical variables related to the speed of the car, rotation of the wheels and other factors, which determine the right instant for the system to actuate). However, other systems in which time constraints are introduced artificially to achieve real-time performance may express their timeliness constraints in a less strict manner.

Quantifying the tolerance towards faulty timeliness performance in a system is a challenging problem which requires a profound knowledge of the system aided by a proper timeliness notion with the ability to express suitable constraints. Section III.2 explores in detail the convenience of a suitable timeliness notion which enables the exploitation of timeliness as a quality metric.

III.2 Notions of Timeliness

The notion of timeliness currently used in WSNs derives from general purpose networks, which at the same time, inherited it from classic RTSs. Thus, the metric is evaluated as the capacity of individual messages to fulfill their associated strict end-to-end deadlines. If a message has not been delivered before the deadline expires it is considered a failure, regardless of the exceeded amount of time. In fact, it is a common practice of many existing protocols to discard messages at one of the intermediate hops if the deadline has already expired (e.g. [He 03]). Even in some cases, the protocol applies an earlier message dropping policy based on estimations made at run-time. However, this practice introduces the risk of rejecting messages based on pessimistic estimations, which otherwise could have achieved their end-to-end delay.

Enforcing the classic notion of timeliness in WSNs has a major drawback, namely, that the assumptions and system models necessary to provide timeliness guarantees in RTSs do not hold in WSNs.

In this section, we analyze the most extended notion of timeliness used in real-time WSNs literature, stressing the limitations and weak points that arise under realistic scenarios. Later on, we introduce a generalized timeliness notion which suits the requirements of WSNs without the need of over-constraining the system models. Finally, we show that the existing methods can be easily adapted to take advantage of the generalized notion without much effort.

III.2.1 Classic Real-Time Notion of Timeliness

The classic real-time notion of timeliness is conceived to provide timeliness guarantees in processor scheduling. As a performance metric, it evaluates timeliness as the capacity to provide a feasible schedule of tasks which fulfill their relative deadlines.
Several scheduling policies and task models exist, but the basic principle can be reduced to the following:

Real-Time schedulability analysis: Given a set of tasks Π –periodic, aperiodic or sporadic– to be executed on a processor, each task π_i has a deadline d_i relative to its release time. The real-time schedulability analysis will find whether a feasible schedule exists, such that all task instantiations of Π (i.e. jobs) are *guaranteed* to fulfill their execution within their release time and relative deadline.

The associated implicit notion of timeliness allows expressing a relative deadline that delimits the execution window for each task. As a consequence, the potential to exploit timeliness is limited to ensuring feasibility based on the execution window. In other words, the available computation time required by each job is guaranteed at the processing unit (i.e. CPU) within the delimited time window. The exact instant of execution is not relevant as long as the constraints are fulfilled, and it may vary for each instantiation.

Despite its simplicity, this notion of timeliness provides an adequate fit to exploit timeliness on classic processor scheduling. A closer look into the system models and their requirements allows to justify this adequacy based on the following observations:

(a) the behavior of CPUs is strictly deterministic; (b) the execution of tasks is local and the overhead due to the system handling can be neglected without harming the final schedule; (c) the probability of malfunction is very low; and (d) the simplicity of the system model allows exact analysis to ensure schedulability guarantees.

Note that the analogy between classic processor scheduling and other systems has its limitations. In some cases, it may be adequate to perform adaptations (e.g. simplifications of system models or addition of assumptions) in order to enable concrete solutions to different problems than schedulers. However, exporting the timeliness notion requires a careful analysis of the new system to ensure consistency with the models and the reality that they represent.

For example, observation b is generally not valid in multi-processor scheduling and observation c is not acceptable in fault-tolerant systems. In real-time networking, the resemblances with the original problem are generally less obvious, although the timeliness notion is successfully applied in some cases (e.g. [Davis 07]). However, the particularities of wireless networks contain system properties that are rarely compatible with the aforemetioned observations.

As a matter of fact, the inherent properties of WSNs allow little space for comparisons: the system does not behave deterministically (observation a); the system is -by definition- distributed, and the effects of the overhead are significantly large (observation b); the probability of interferences, contention due to collisions, and message losses are high (observation c); and, given the complexity of the system and its dynamic behavior, exact analysis is not feasible unless important and restrictive simplifications are done (observation d).

This being said, the timeliness notion has been used -and still is- in the vast majority of real-time solutions for WSNs. The consequence is that in most cases, restrictive assumptions are needed to grant the applicability of the method. Examples of such are:

- Error-free medium channel without interferences, transmission errors, or retransmissions at MAC level (e.g. [Mizanian 09]);
- Constant bandwidth between any two nodes (i.e. durable strong links) (e.g. [Sahoo 07]);
- Perfect clock synchronization between nodes;
- Fixed routes, known hop-distance, or constant routing trees during network lifetime (e.g. [Ergen 07]).

These assumption, among others (see section II.5), may be acceptable in specific scenarios. However, their enunciation imply the loss of generality, which is often not explicitly acknowledged and leads to misinterpretation.

III.2.2 A Generalized Notion of Timeliness

In this chapter, we explore a different approach to achieve a better alignment between the network properties and the notion of timeliness. Instead of restraining the models to fulfill idealized timeliness properties, we propose to adjust the concept of timeliness to suit the particularities of WSNs. The motivation for a suitable timeliness notion is listed following:

- The capability of the timeliness notion to express constraints should not encourage applications to demand unfeasible degrees of performance that the network is not able to provide. Hence, given the unfeasibility of WSNs to guarantee strict deadlines, applications should not be requested to express constraints at individual messages.
- Given the variability in the timely response, a notion of timeliness expressing only success or failure (i.e. deadline met or not) is of limited value to WSNs. Rather, a continuous function to embody the level of conformance –or deviation– with respect to the timeliness constraints would contribute in a more significant way to elaborate meaningful timeliness protocols.
- The enforcement of end-to-end timeliness is limited and variable at run-time. A meaningful notion of timeliness should account for this variability as well as provide means to express and evaluate it at run-time the level of fulfillment.
- A timeliness notion should be able to express not only explicit constraints but also provide means to reflect the current state of the network, enabling the exploitation of network trade-offs to adjust to feasible levels.

Taking into consideration these aspects, we propose a *generalized timeliness notion* that allows expressing timeliness constraints in accordance with the particularities of WSNs. The generalized notion does not restrain the applicability of timeliness protocols to specific scenarios. However, if necessary, it stills provide means to express strict constraints in particular cases where they are reasonable.

Timeliness constraints are expressed in terms of a *sequence of messages* instead of individually. This avoids conflicts with the non-determinism associated to individual delivery delays while it still provides means of expressing meaningful constraints. A sequence of messages is a consecution of messages with one identifiable characteristic. Unless otherwise stated, in the remaining of this thesis we will refer to a sequence of messages as a series of messages following the exact same path from source to destination (i.e. same end-to-end route). However, the interpretation is intentionally left open as it can accommodate to the end-to-end requirements generated at different networking layers (e.g. application, transport). Other possible uses include:

- series of messages with same source and destination, but not necessarily following the same path;
- messages arriving at a common sink independently of their source or end-to-end path;
- series of messages sharing a portion of a path (i.e. sub-path) for a given time;
- any combination of the above with the additional constraint of belonging to a particular message kind (e.g. “urgent messages”, “alarm”).

The generalized notion does not only allow the expression of requirements in terms of timeliness constraints, but also provides means to express the current timeliness performance at run-time. This is particularly meaningful in large WSNs as well as in deployments in which the conditions are not known at design-time, or in general when the uncertainties of the environment do not allow predefined constraints.

The generalized notion of timeliness is composed of the following parts:

1. A time interval (t_i, t_j) with $t_j > t_i \geq 0$, which delimits the target end-to-end transmission interval for a sequence of messages;
2. The level of confidence –as a constraint or an observation– on successful end-to-end transmissions within the interval, expressed by means of a probability $0 \leq p \leq 1$.

Note that the notion itself neither compels nor defines a particular method to exploit timeliness. The notion solely provides means of expression that enables multiple opportunities towards meaningful timeliness approaches.

For a better understanding of the generalized timeliness notion, the following example introduces a use-cases showing its potential. An extended application with a proposed algorithm for the estimation of end-to-end latency is described in section III.3.

III.2.3 Use-Case Example

The following use-case describes the scenario of the example:

Use-case: Assume a medium size WSNs. All existing nodes in the network perform the common task of monitoring certain environment variables. Under normal conditions, the values are processed and periodically transmitted to a data collector node (i.e. sink) at a low frequency. As long as the readings remain under a given bound, timeliness performance is not relevant and a low energy consumption profile is prioritized. However, if one of the nodes detects sample reading beyond the bound, it increases its monitoring and transmission frequency. In this mode, timeliness is prefered over energy consumption.

A sample application following this behavior is the monitoring of air quality in areas exposed to contaminating agents (e.g. chemical factories). Under normal conditions, regular sensing at long intervals suffices to provide an overview of the air quality. However, if an escape of polluting elements occur, the timely transmission of sensor readings become crucial. Note that for the sake of simplicity, we do not consider life-critical emergencies but rather situations in which timely reports are needed for contention purposes.

From the description of the use-case that two modes are implicitly defined: (a) *normal mode*, with low frequency monitoring and low energy consumption scheme; and (b) *alert mode*, with higher frequency monitoring and prefered timeliness communications.

With these constraints, it seems obvious that the application must operate in two different modes matching the two use-cases modes. From a timeliness point of view, mode a) does not introduce any challenge. On the contrary, mode b) introduces timeliness constraints which must be confronted.

Current Approach – Classic Timeliness Notion:

Following a classic real-time approach, these constraints are introduced in the system in the form of message deadlines. The selection of values can be done in multiple ways, e.g. expertise on the field, performance figures extracted from simulation runs, or analytical deduction from the problem constraints. However, if the decisions are taken either based

on expertise or through system models, there are a number of considerations that cannot be made at design time and affect in great measure the timeliness performance of the system. Namely:

- the environment conditions at run-time (e.g. radio connectivity due to interfering noise, high traffic due to multiple nodes reporting events, etc);
- the topology of the network may have been altered, or even continuously reorganized at run-time (e.g. reconfiguration of connected nodes, mobility, dying nodes, etc);
- routing paths from source to sink may be unknown at the instant of the event.

A conservative estimation of parameters increases the probability of satisfying the timeliness constraints. However, since the worst case scenario is not bounded, guarantees are unfeasible, and therefore at run-time end-to-end deadlines may or may not be fulfilled.

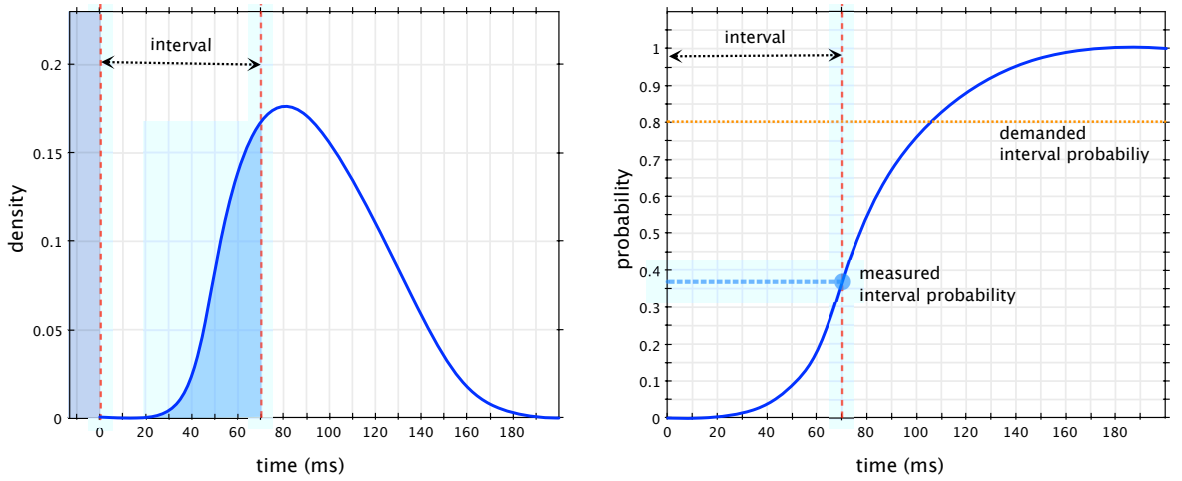
Run-time adaptations of the network configuration (e.g. protocol parameters) can improve the timeliness performance at run-time. However, the information provided by the classic timeliness notion poorly contributes to explore the adequate trade-offs. If timeliness constraints are satisfied, it is not trivial to decide whether a lower energy consumption level will incur in deadline violations. On the hand, if the constraints are not fulfilled, it is difficult to assess amending actions and to obtain the appropriate set-points of operation.

Proposed Approach – Generalized Timeliness Notion:

With the generalized timeliness notion, the selection of adequate values for the timeliness constraints is more flexible. At first, initial end-to-end time intervals are chosen following a similar approach as with the classic notion. However, unlike the boolean information expressed in the classic notion, the inclusion of an interval probability allows expressing the tolerance of the system to unsatisfied constraints. Contrasting both parameters with the run-time end-to-end delays distribution provides valuable information regarding the deviation between the current and aimed performance.

If the timeliness constraints are not satisfied, the probability of messages arriving within the interval is contrasted with the run-time measurement. Unlike using the classic notion, the gap between the performance observed at run-time and the one demanded at design-time is now measurable. Figure III.2 depicts one such situation. Note that with the classic timeliness notion it is also possible to collect statistics with respect to deadline misses with respect of a given interval. However, the information provided by the end-to-end distribution allows for dynamically obtaining the probability of any such interval as well as the interval for which a given probability is achieved.

Either if the system decides to perform network-level trade-offs or re-adjusts the constraints to feasible levels, the generalized timeliness notion reflects the adaptations in a measurable manner. The contrast between the aimed timeliness and the run-time performance introduces a feedback channel for the network trade-offs that make possible a fine coarse tuning of the operational set-points. In particular, it is possible to apply lower set-points for other quality metric while keeping track that the timeliness



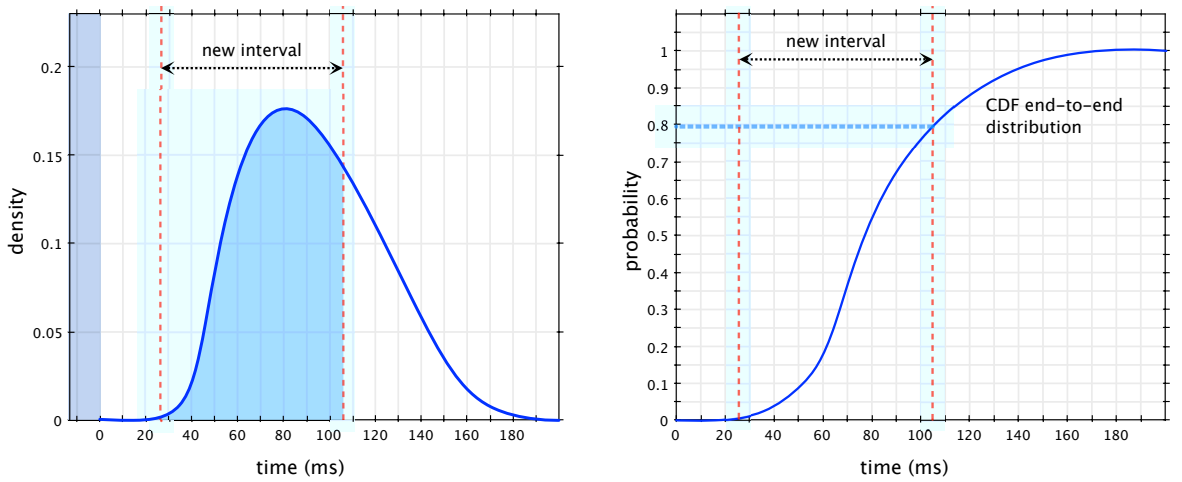
(a) *PDF of the run-time distribution of end-to-end delays*

(b) *CDF of the run-time distribution of end-to-end delays*

Figure III.2: *Example of unsatisfied timeliness constraints with the generalized notion.*

performance remains within acceptable levels.

Figure III.3 depicts a simple adaptation consisting of increasing the bounds of the target interval to achieve the aimed probability of success. In this case, no other actions are taken to improve the timeliness performance (i.e. trade-offs to improve timeliness). However, being able to estimate the probability for bounded end-to-end response time will suffice in many application scenarios to provide satisfactory timeliness results.



(a) *PDF of the run-time distribution of end-to-end delays*

(b) *CDF of the run-time distribution of end-to-end delays*

Figure III.3: *Example of adaptation to the current network status with the generalized notion.*

The adaptation depicted in III.3, shows that the left-side limit for the end-to-end

interval is shifted to the right. While this does not necessarily make sense as a timeliness constraint –i.e. constraining faster transmissions–, it does indeed provide valuable information to the application with respect to the minimum transmission time. Thus, any expectation of achieving message transmissions below this limit have a probability of success of approximately 0.

III.3 Probabilistic estimation of end-to-end latency

Practical applications of the generalized timeliness notion depend on the capability of calculating the run-time distribution function of the end-to-end delay. Unfortunately, a purely analytical deduction of the distribution is not feasible –unless based on restrictive abstraction models– due to the non determinism of WSNs. Nevertheless, in this section we describe a good run-time estimation method with low demand of resources.

The end-to-end delay experienced by any message transmitted across a routing path rp can be decomposed into the individual forwarding delays originated at each node $n \in rp$. Therefore, the estimation of the end-to-end distribution is done in two steps. First, we a analyze and characterize the distribution of the forwarding latency at each intermediate hop. Second, we perform an approximate composition of the end-to-end distribution for the entire end-to-end path. This procedure is continuously repeated at run-time, hence adapting to the changing network status.

III.3.1 Notation

A WSN is represented as a graph $G(N, L)$ formed by a set of nodes N and a set of *single-hop links* L. Two nodes $n_i, n_j \in N$ are *directly connected* at a given time if there is a link $l \in L$, $l = (n_i, n_j)$ such that n_i and n_j can send and receive messages from each other. For the sake of simplicity, the time instance of variables that change over time are only represented when they are significant to the analysis.

$S \subset N$ is the subset of sinks. Sinks may outperform nodes with respect to resources and energy availability.

A *(routing) path* rp is a sequence of nodes $(hop_1, hop_2, ..hop_{q-1}, hop_q) \in N$ without repetitions such that each pair $l_i = (hop_i, hop_{i+1}) \in L$, thus providing a *multi-hop link* between n_1 (source node) and n_q (destination node). The length of a path is equal to its number of links ($|rp_{hop_1,hop_q}| = q$).

We notate Φ as the effective end-to-end delay of a message (given an end-to-end path).

III.3.2 Calculation of one-hop forwarding latency

Let $hop_1, hop_2 \in N$ be two nodes such that $\exists l = (hop_1, hop_2) \in L$. Then, we define D_{hop_1} as the *Random Variable (RV)* which characterizes the latency δ experienced by a message being forwarded from hop_1 to hop_2 and $F_{D_{hop_1}}$ as the *Cumulative Distribution Function (CDF)* such that $F_{D_{hop_1}}(\varepsilon) = P(D_n \leq \varepsilon)$; the probability that hop_1 introduces a delay of at most ε in forwarding a message.

The forwarding latency δ is defined as the time interval between the message arrives at a node t_{in} –either because the application layer sets a new message to be sent, or because the MAC layer receives a message which has to be forwarded–, and the reception by the sender of an acknowledgment from the receiver t_{ack}. Note that this calculation is pessimistic as it introduces the additional time due to the transmission of the acknowledgment. For certain MAC protocols this might be negligible, but otherwise it can be estimated as a constant $\varphi \geq 0$. Therefore,

$$\delta = t_{ack} - t_{in} - \varphi \tag{III.1}$$

The calculation of δ at hop_i is computed every time it forwards a new message. The sequence of values over time (i.e. $\delta^0, .., \delta^k$), represents the samples of the RV D_{hop_i}.

Note that in section III.3.3 we introduce a cumulative method to avoid storing the whole sequence of values in the node.

III.3.3 End-to-end latency distribution

We describe the analysis of the end-to-end distribution in two steps: first, we analyze the simple case with one single hop; and later on, the general case with $|rp| > 1$.

Simple case: one hop

In the simplest case, a message is forwarded by a single node. Let $sender \in N$ be the sender node and $sink \in S$ the receiver. Let $l = (sender, sink)$ be the link between them. Thus, the end-to-end latency is equal to the forwarding latency of the only hop, which is δ_{sender}. By definition, the end-to-end delay distribution D_{e2e} equals the forwarding distribution D_{sender}, since $|rp| = 1$ with $rp = \{l\}$

The distribution function of D_{e2e} depends on many factors which are generally not controllable: e.g. link quality, environmental noise, and most relevant: the underlaying MAC protocol. These, unfortunately, complicate their characterization as it is not feasible *a priori* to extrapolate the CDF. However, we can easily estimate the mean value and variance of the forwarding latency of D_{sender} $(\mu_{sender}, \sigma^2_{sender})$. The estimation of these values at run-time provide a rough indicator of the end-to-end distribution for this particular case.

The sample mean $\bar{x}$ and sample variance s^2 are good estimators of μ and σ^2. We use the *Exponential Weighted Moving Average (EWMA)* [Burgstahler 02], [Goldoni 08] to achieve low-resource-demanding run-time estimations (Equations III.2 and III.3). EWMA generates two new variables $\bar{x}^*_{it}$ and $s^{2\,*}_{it}$, which are updated at each iteration (noted as it) and approximate the mean and variance.

$$\bar{x}^*_{it} = \alpha\delta_{it} + (1-\alpha)\bar{x}^*_{it-1} \tag{III.2}$$

$$s^{2\,*}_{it} = \alpha(\delta_i t - \bar{x}^*_{it})^2 + (1-\alpha)s^{2\,*}_{it-1} \tag{III.3}$$

Equations III.2 and III.3 are computed at each hop every time a message is forwarded through it, hence updating the local state information.

The parameter α $(0 \leq \alpha \leq 1)$ is set to weigh the actual measurements with respect to the past trend. The discussion on the selection of its value follows in section III.3.5.

General case: k hops

The end-to-end distribution of a multi-hop path is obtained by the composition of the distributions of each intermediate hop. Despite the aforementioned unfeasibility in calculating the exact end-to-end distribution, an approximation can be obtained with help of the *Central Limit Theorem (CLT)* [Bulmer 67].

Central Limit Theorem: The classic CLT states that the sum of a number (k) of RVs with approximately the same distribution, non-negative and mutually independent tends to a *Normal* distribution $N(\mu, \sigma^2)$.

The motivations to accept the conditions of the classic CLT, and in particular the interdependency of the RV, are discussed in Section III.3.5.

Note that despite the CLT is commonly applied to larger number of samples ($k > 20$), its approximation to the Normal distribution is already noticeable within the sum of few RVs. An argumentation about good approximations for small sums of RVs is given in [Korkmaz 03].

Hence, we define the RV D_{rp} characterizing the end-to-end distribution of path rp as:

$$D_{rp} = \sum_{\forall l \in rp} D_l \tag{III.4}$$

and,

$$F_{D_{rp}}(\tau) = P(D_{rp} \leq \tau) \tag{III.5}$$

And the parameters μ_{rp} and σ^2_{rp} of D_{rp}:

$$\begin{aligned} \mu_{rp} \approx \bar{x}_{D_{rp}} &= \sum_{\forall l \in rp} \bar{x}^*_{l\,D_l} \\ \sigma^2_{rp} \approx s^2_{D_{rp}} &= \sum_{\forall l \in rp} s^{2*}_{l\,D_l} \end{aligned} \tag{III.6}$$

According to the CLT, the probability introduced in Equation III.5 converges to:

$$\begin{aligned} F_Z(\tau) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\tau} e^{\frac{y^2}{2}} dy \\ Z &= \frac{D_{rp} - \mu_{D_{rp}}}{\sigma_{D_{rp}}} \end{aligned} \tag{III.7}$$

III.3.4 Example of End-to-End Latency Estimation

The following example based on simulations illustrates the applicability of the end-to-end estimations. The simulation set-up follows the description in VI.1.4. The sum of parameters estimated at each hop of a routing path is defined as the pair $\Delta_{\bar{x}}, \Delta_{s^2}$. Under

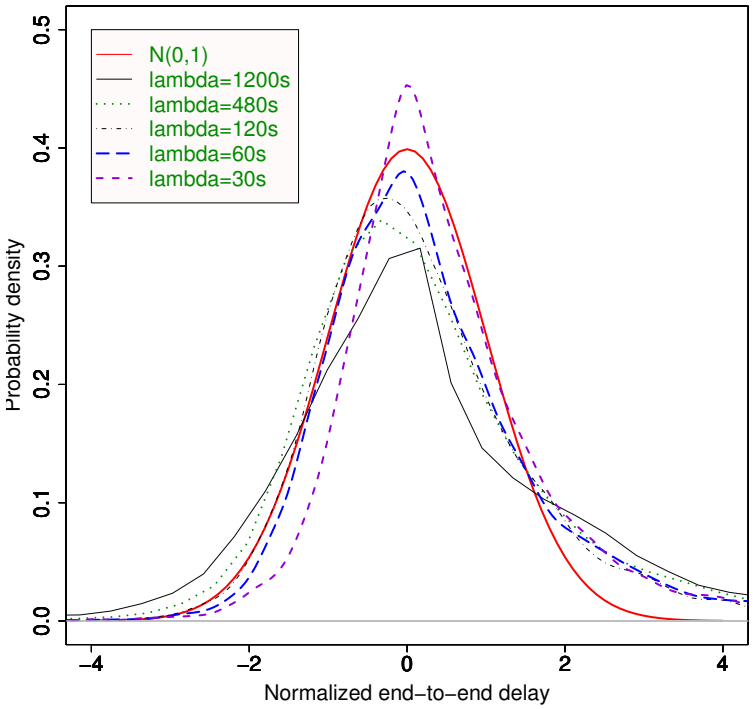


Figure III.4: *Simulation results for scenario with $|rp| = 5$ and traffic parameter λ. Showing PDFs of estimated distributions compared to the Normal distribution [N(0,1)].*

static network conditions (i.e. assume a network snapshot), the distribution $N(\Delta_{\bar{x}}, \Delta_{s^2})$ is representative of the effective distribution of the end-to-end latency of that path.

However, both the estimated and the real distributions are dynamically changing. The network conditions are different every time that a message is being forwarded – its distribution is not constant for the entire network lifetime–. Similarly, each time a message is forwarded, the forwarder node updates its parameters $\bar{x}^*$ and s^{2^*} which induces a change in the estimated distribution.

Hence, for each instance *it* of a message going through the analyzed path we capture an effective end-to-end delay Φ^t (i.e. measured in the experiment) and a set of parameters $\Delta^{it}_{\bar{x}}, \Delta^{it}_{s^2}$ (i.e. estimations), which are not directly comparable to those originated by previous or following messages. Each pair of estimated and measured parameters can be compared individually but they cannot be taken as samples of the same RV.

We define two tests to evaluate the accuracy of the estimations:

Test 1: Normalize each sample of the effective end-to-end distribution to the standard Normal distribution $N(0, 1)$. Given,

$$X \sim N(\mu, \sigma^2)$$

then,

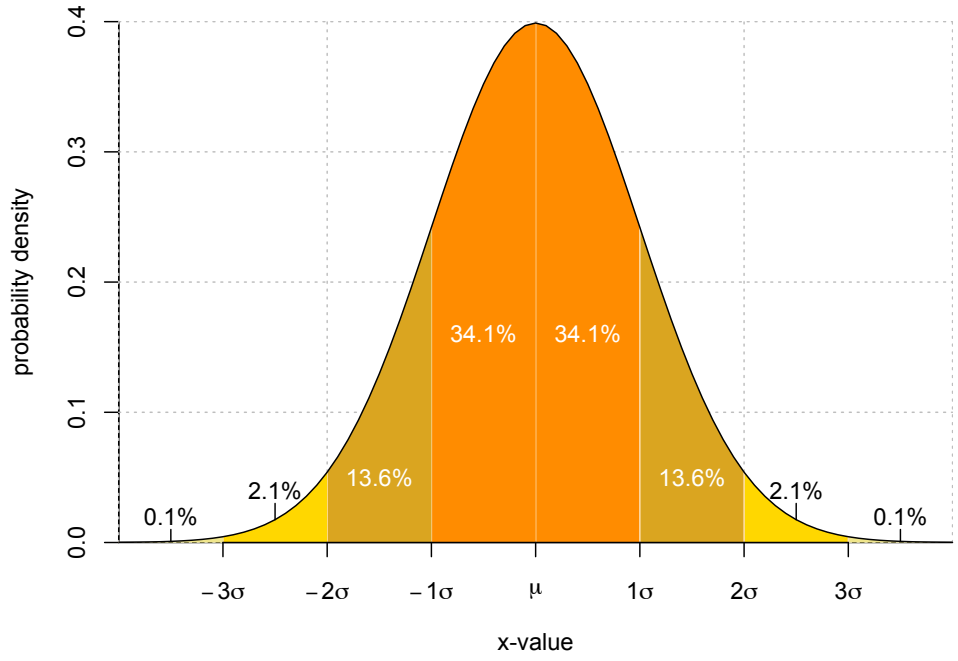


Figure III.5: *Standard deviation intervals of the Normal distribution* $N(0,1)$

$$Z = \frac{X-\mu}{\sigma}$$
$$Z \sim N(0,1).$$

This way, instead of comparing each individual sample to a $N(\mu,\sigma)$ with different parameters, we can compare all samples against a $N(0,1)$. The expectation is that the distribution of normalized samples approximates a $N(0,1)$.

Test 2: Compare the number of "hits" of each interval determined by the distance σ from the center point (μ). This is known to be around 68%, 27%, 4.2% and 0.2% respectively for the intervals $I_1 = (-\sigma,\sigma)$, $I_2 = (-2\sigma,-\sigma) \cup (\sigma,2\sigma)$, $I_3 = (-3\sigma,-2\sigma) \cup (2\sigma,3\sigma)$ and $I_4 = (-\infty,-3\sigma) \cup (3\sigma,\infty)$ (see Figure III.5[1]). If the estimated distribution is accurate, the number of samples falling in each of these intervals should approximately follow these proportions.

Example Simulation

The example simulated traffic messages from a sender node hop_1 to a sink hop_q with the interference of cross-traffic coming from neighbor nodes as depicted in Figure III.6. The

[1]Figure generate in R, based on code from Petter Strandmark.

results show ten simulation runs with different traffic parameters for a routing path of 5 hops. For a complete scenario description and extended results refer to chapter VI.

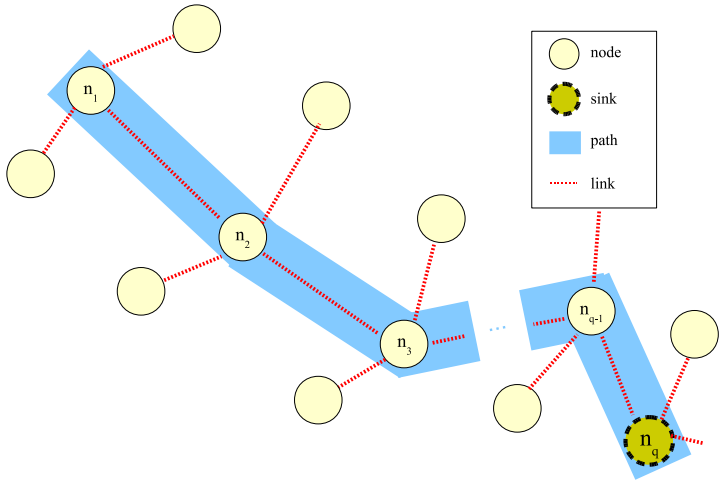


Figure III.6: *Simple simulation scenario*

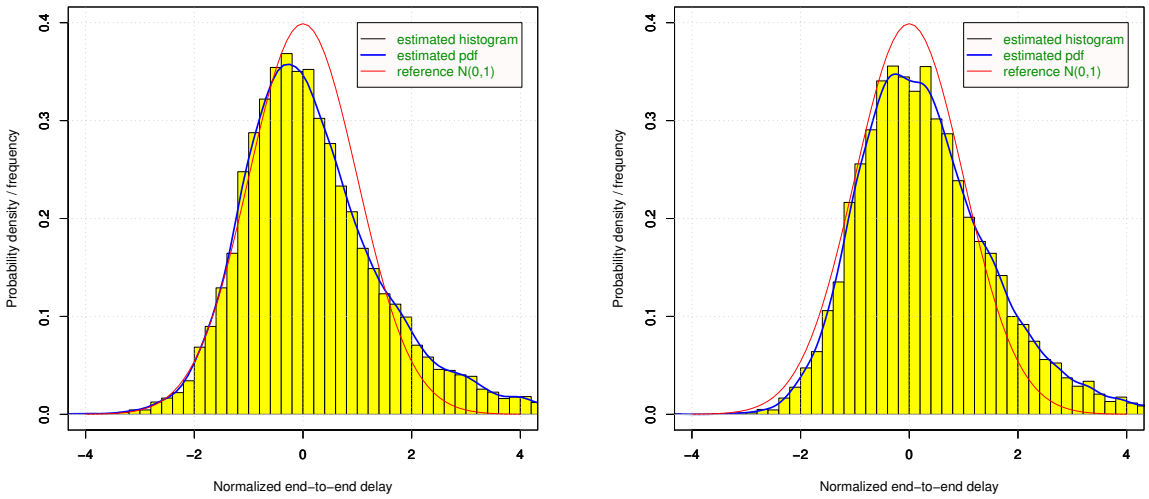
Note that for the sake of simplicity, the process of building the routing path (i.e. routing tree) is not considered in this example. The purpose of the simulations is to evaluate the validity of the method and illustrate the process to obtain the end-to-end delay distribution.

Figure III.7 shows the histogram of the distribution for the case of $|rp| = 5$. The distributions are normalized and compared to the Probability Density Function (PDF) of the standard Normal $N(0, 1)$. Note that the depicted graphics are cropped at the interval $(-4, 4)$.

At first sight, two questions arise: the difference between the two curves at the central point and the larger tail on the right side. Both effects are related to each other and can be explained by the nature of the experiment measurements. In fact, the values represented come from measured end-to-end delays. This necessarily introduces a tail effect, as there is a clear limit on the possible values from the left side (i.e. time delays cannot be negative) but none on the right side.

With respect to the range of absolute values, having a mean sample value of $6.5ms$ very few messages achieved a delay less than or equal to $2ms$ and the distance between the minimum value and the mean is approximately $5ms$. However, on the right side, this distance goes up to around $34ms$, with a maximum value close to $40ms$.

Note that the α parameter on the EWMA is, to some extent, responsible of this effect. A lower α acts as a filter for higher sampled values and hence, reduces the tail on the right side. However, this also affects the sample variance s^{2^*} as the estimated values get closer to each other. Thus, low values of α introduce a distortion on the estimated distribution which results in "thinner" curves. On the other hand, higher values of α reduce the smoothing effect of the EWMA but produce a more accurate estimation of the sample variance. This is reflected on the peak of the estimated distribution, although,



(a) *Normalized histogram and estimated pdf vs* $N(0,1)$*, for* $|rp| = 5$

(b) *Normalized histogram and estimated pdf vs* $N(0,1)$*, for* $|rp| = 10$

Figure III.7: *Example of estimated distribution after simulation.*

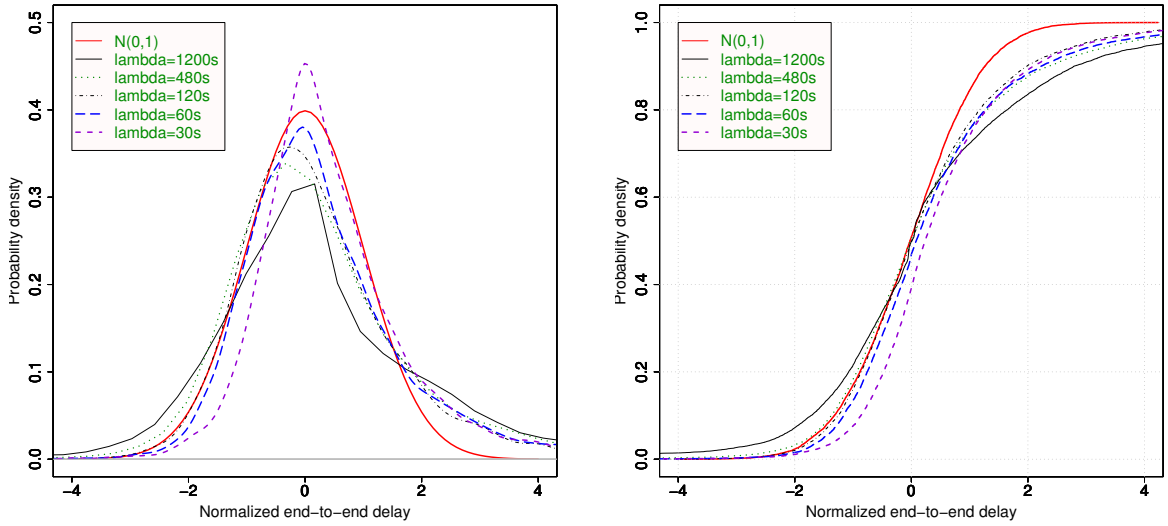
at the same time, produces thicker distribution shape. Based on experience of previous simulations [Serna Oliver 09a], we chose $\alpha = 0.9$, which has provide accurate estimation without introducing excessive distortions on the final distributions.

Figure III.8a and shows the probability density and cumulative distribution function (i.e. PDF and CDF) of all cases with $|rp| = 5$ and variations in the cross traffic (λ parameter).

Observe that accuracy increases proportionally to the cross-traffic parameters. This is due to the fact that the higher the amount of messages going through the network, the more frequently intermediate nodes refresh their local estimations. In other words, if the traffic is too low, the estimated values at the arrival of a message loose accuracy by the time the next message is received.

In figure III.8b, it is noticeable the "lower peak" described before from the point of view of the estimated probability. The higher part of the curve is visibly below the reference curve, which means that the estimation becomes pessimistic (i.e. the method will predict a lower probability for delays above the expected end-to-end delay). However, the same does not happen, except for the case of very low traffic, in the lowest part of the curve. This means that the estimated probability for end-to-end delays below the expected value do not over-estimate the capacity of the path.

Table III.1 present the results for the second test with the reference to the standard Normal in brackets. Again, the tail effect is visible as the interval I_4 receives significantly more hits than expected. Similarly, interval I_1 reflects a lower percentage of hits, which agrees with the previous figures.



(a) *Estimated pdfs vs N(0,1), with* $|rp| = 5$ **(b)** *Estimated cdfs vs N(0,1), with* $|rp| = 5$

Figure III.8: *Results of simulation runs with different configuration parameters.*

λ N(0,1)	I_1 (68%)	I_2 (27%)	I_3 (4.2%)	I_4 (0.2%)
30	66.5% (-1.5)	21.5% (-5.5)	6.8% (+2.6)	5.2% (+5)
60	62.2% (-5.8)	24.6% (-2.4)	7.1% (+2.9)	6.1% (+5.9)
120	61.1% (-6.9)	27.1% (+0.1)	7% (+2.8)	4.8% (+4.6)
480	53.3% (-14.7)	27% (=)	8% (+3.8)	7.7% (+7.5)
1200	50.8% (-17.2)	25.6%(-1.4)	11.9% (+7.7)	11.8% (+11.6)

Table III.1: *Percentage of hits per* σ*-interval with path length* 5*. In brackets, deviation with respect to* $N(0,1)$*.*

III.3.5 Additional Notes

For the sake of clarity, some relevant details were omitted from the description of the end-to-end estimation method. In the following paragraphs two of them are discussed. A complete description of the scenario as well as the simulation environment and parameters are detailed in chapter VI.

Selection of an appropriate value α for EWMA

The exponential weight α controls the smoothing factor in Equations III.2 and III.3. Lower values of α increase the stability of the measurements as they smooth the variation due to small fluctuations with respect to the averaged value. This is the desired effect to avoid imprecisions due to fluctuations on the sequence of measurements. On the other

hand, large values of α tune the equations such that they adapt to changes and forget the past values quickly.

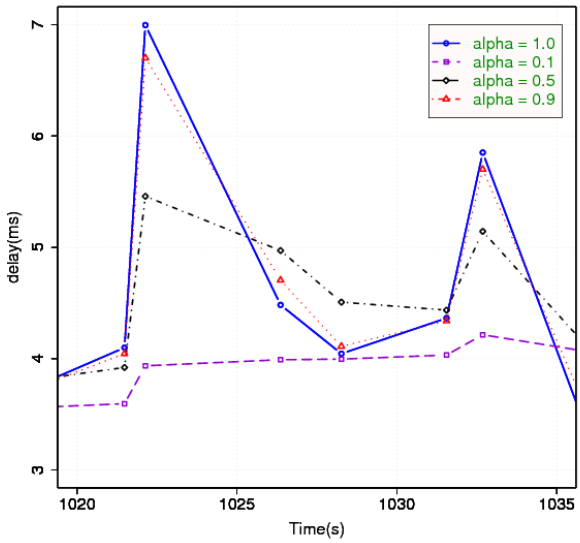


Figure III.9: *Hop latency with different values of α.*

Figure III.9 shows the evolution of $\bar{x}^*$ for different values of α during a short interval of time extracted from the simulation results of the previous example. Despite the benefit of the smoothing qualities of a low α for the calculation of $\bar{x}^*$, it is not the case for s^2 (Equations III.2 and III.3). In the latter case, the variability of values needs to be captured as it plays an important role in the estimation of the final distribution.

After performing extensive simulations [Serna Oliver 09a], we observed that it is more beneficial to reflect the variability of the data in order to obtain accurate s^2, than to obtain a smoother value of $\bar{x}^*$. Nevertheless, EWMA still proves effective to reduce the effect of occasional overestimations of the forwarding delay – due to i.e. radio anomalies. The experiments show that a large value of α close to 0.9 produces acceptable results for most cases.

Assumptions on the conditions of the CLT

Message dependencies (i.e. $E[V_a, V_b] \neq E[V_a] \cdot E[V_b]]$) might appear under certain circumstances. For instance, dependencies happen when a message m_1 causes additional delay to a message m_2 which message m_2 would not experience in isolation.

Other reasons for dependencies to happen are related to medium access policies (e.g. back-offs) and buffer constraints (e.g. messages dropped due to buffer overflows). However, the typically low traffic of WSN reduces the probability of these conditions to happen. Nevertheless, it is possible that during intervals of time with higher traffic load the dependencies are reflected in the estimations.

Literature presenting alternative variants of the CLT which relax the conditions of inter-dependency exists (e.g. [Birkel 93]). However, the prerequisites of these reformulated versions of the theorem impose new conditions which cannot be guaranteed without detailed analytical models of the network.

Hence, dependencies are accounted for at run-time as the delay distribution introduced by each hop is continuously estimated with the current timeliness performance. Thus, the estimated parameters for a certain hop as well as the composed end-to-end delay distribution vary for different network loads.

The spatial distribution of nodes and the typical low throughput of WSN minimize the effects of transmission dependencies, and even in situations of high load, these are not significant to the desired accuracy for the estimations. After performing simulations [Serna Oliver 09a], we observed that the deviations due to dependencies are not relevant in the general case as they may only arise in situations of very high network saturation.

III.4 Chapter summary

This chapter overviewed the essential flaws in application of the classic timeliness notion in current real-time approaches for WSNs. In contrast to the classic notion, it presents a generalized timeliness notion providing means to express meaningful timeliness in typical WSNs without restraining the feasibility of methods.

As a proof of concept, the chapter introduces a probabilistic algorithm for the estimation of end-to-end latency of message transmissions in WSNs. The estimation is based on the generalized timeliness notion obtaining the distribution function of end-to-end routing paths.

Chapter IV

Timeliness Trade-offs

The ultimate goal of a real-time method for Wireless Sensor Networks (WSN) is to provide timely responses to selected events while fulfilling predefined Quality of Service (QoS) requirements. However, defining the general objectives determining these requirements is not a trivial task. The generalized timeliness notion presented in chapter III opens a new range of possibilities exploring QoS trade-offs as a means to fulfill timeliness constraints while easing the specification of feasible objectives.

The new notion based on the distribution of end-to-end delays makes possible defining flexible requirements allowing local and global run-time adaptations, which contribute to enhancing the timeliness performance at particular moments. These adaptations may allow pursuing realistic end-to-end timeliness based on adaptive QoS requirements, but proper mechanisms to enforce the necessary adaptations as well as hooks allowing QoS trade-offs must be established.

Specifying timeliness objectives in a WSN overcoming the potential clash between feasible requirements and the limitations imposed by protocols and scenarios is not trivial and may still require a certain amount of individual analysis for each deployment. In particular, exercising trade-offs between timeliness and other QoS metrics introduces a number of potential adaptations that real-time methods can exploit achieving satisfactory QoS levels. The chapter is organized as follows:

Section IV.1 introduces the elemental considerations defining feasible timeliness objectives enabling meaningful QoS methods. The definition of these objectives and the establishment of instruments to enable them are as important as providing means evaluating the qualities of the target system.

Section IV.2 analyzes the possibilities provided by the software and hardware platforms to exploit local trade-offs.

Section IV.3 explores the definition of global timeliness trade-offs –in new or existing real-time methods– taking into account the necessary parts having a significant impact on the real-time performance. The analysis is centered on the aspects that influence the applicability of the generalized timeliness notion and the exploitation of QoS trade-offs. In doing so, the chapter explores the definition of abstract models for the analysis of metrics as well as the impact of selected parameters, both from a global and local domain perspective.

Finally, the chapter concludes with a summary.

IV.1 Timeliness Quality of Service

For systems where a single metric dominates the perception of Quality of Service (QoS), a simple optimization analysis leads to the optimal configuration of set-point providing the maximized value for that metric. However, when the metrics of interest are several it is often the case that a simple optimization criterion cannot satisfy the optimal values for all metrics. In these cases, a multi-metric trade-off analysis produces a set of valid configurations, each of them being optimal regardless of the optimal value corresponding to each metric in isolation.

This chapter explores the application of multi-metric QoS analysis and in particular of QoS trade-offs as means to approach timeliness in Wireless Sensor Networks (WSN). The analysis focuses on trade-offs involving timeliness and at least one other metric of interest, which –given the characteristics of WSNs– will in most cases be related to the usage of energy. However, the analysis is valid for any other number of metrics. The first step, consisting of defining an adequate QoS metric providing a qualitative evaluation of timeliness, has been already addressed in the previous chapter, when introducing the generalized timeliness notion. This section explores the definition of objectives leading to feasible timeliness approaches.

IV.1.1 Definition of Timeliness Objectives

The particularities preventing strict real-time methods from being applied into WSNs are at the same time a valid motivation to exploring QoS as a means to approach timeliness. In particular, the tight relation between the cost in terms of resources and the achieved performance, suggests a significant potential in the exploitation of QoS mechanisms.

Aligning the QoS requirements with the network capacity of fulfilling them is a fundamental constraint that must be taken into account by any real-time method. In terms of QoS requirements, it is important to consider the constraints imposed at run-time by the scenario of deployment. Based on these constraints, the strictness of feasible real-time requirements may vary and specific objectives that are not achievable in the general case may become possible.

In most cases, the scenario itself plays a fundamental role in determining the real-time capacity of a deployed system. Thus, incorporating details of the final deployment is a crucial step defining the objectives of a real-time method. Not doing so may lead to misleading goals as those described in section II.5 providing a false sense of achievement with respect to timeliness performance.

For example, taking a single-hop deployment of a static WSN without strong energy constraints, it is possible to imagine that bounded times for message delivery latencies may be achievable. One such scenario fits in the characteristics of an in-door surveillance system with nodes connected to the main power line and statically placed along the surveilled area. With a careful selection of protocols and system components, the combination of unconstrained transmission capacity and the presence of a powerful gateway in this single-hop set-up may allow nodes delivering messages in a timely manner.

However, applying the exact same system configuration into a large multi-hop scenario suffering of strong energy constraints may lead to unpredictable delays, reducing drastically the chances of obtaining time-bounded communications.

Unfortunately, considering variables of one particular deployment is not always practical for the definition of general networking stacks and components. Creating *ad-hoc* solutions may provide the best performance to particular scenarios, but in most cases the benefits of such specific construction do not compensate the efforts involved in designing, developing, and testing the final system. The compromise between the flexibility of a method fulfilling timeliness requirements and the generality of the solution adapting to any deployment scenario is not obvious and needs careful analysis.

There is no automated *recipe* to find the optimal definition of objectives given a network set-up, or elseways a directive providing the optimum deployment set-up satisfying a set of real-time objectives. Nevertheless, the following list profiles a guideline applying to most WSNs which are worth considering.

Assumptions – Some of the existing real-time methods do not directly expose the limitations that result from the assumptions they are based on, although a close look to their constitution and the necessary conditions for their implementation establish clear limitations to defining feasible timeliness objectives. Some of the misleading assumptions that should be avoided or carefully considered in large deployments are listed in section II.5.

QoS Metrics – Using proper quality metrics reflecting the performance of evaluative criteria is crucial to achieving significant results. Although it may seem an obvious remark, these metrics must be chosen with consideration to the variables that they measure as well as their impact in the desired qualitative performance. It is often the case that existing real-time methods produce optimal solutions measured by inaccurately defined metrics. A similar analysis of that presented in chapter III with respect to the timeliness metric may aid defining appropriate QoS metrics for additional system variables.

Compatibility – Most real time methods are often tested independently and their performance evaluated based on ideal conditions. However, complete solutions built of individual components –e.g. networking stack– require a global evaluation that may question their compatibility. Existing frameworks and middleware, explored in section II.3.1, may offer solid approaches although their flexibility to exploit QoS trade-offs are generally more limited.

Cross-optimizations – Fruit of the interaction between different layers and components, new cross-layer and cross-component optimization chances may appear. Exploring and taking advantage of these options can only benefit the construction of solid timeliness solutions, although the risk of losing generality exists. Explicit cross-layer designs exposing selected optimization hooks that may or not be exploited at other layers are certainly promising but require additional efforts.

IV.1.2 Metrics and Parameters

The definition of network trade-offs requires a meticulous analysis of QoS metrics and parameters. The domain at which these metrics are defined affects the ambit of applicability of the related trade-offs. Hence, system-wide metrics must be defined in oder to obtain global trade-offs.

The range of possible metrics in WSNs goes from those evaluation the energy efficiency of radio communications to metrics quantifying the performance of network protocols or full network stacks. Some metrics may be specific to particular application behavior, such as quantifying the effects of multi-level algorithms or the scalability of a particular method. Others, characterize properties of the entire network, such as the overall reliability or the network security.

The importance of choosing a consistent set of metrics coinciding with the evaluation goals is crucial. In particular, there is a tight relation between the selected metrics and the parameters enabling the trade-off analysis. The consideration of which parameters enter in the trade-off analysis is guided by two complementary criteria. On one hand, the impact that each parameter produces on the selected metrics is taken into account. Parameters that do not contribute to changes in the metric at all or their impact is insignificant are not interesting for the present analysis. On the other hand, the availability or suitability of the system to adjust their settings defines the degree at which these can be operated. The nature of each parameter as well as the level at which the analysis is conducted defines whether they are controllable or not. Those which cannot be controlled cannot contribute to the definition of trade-offs, even though they may still have a significant impact on the metrics.

The number of controllable parameters present in the hardware and software platforms vary from one deployment to another. In particular, the networking stack provides important hooks to those parameters with the highest impact on timeliness. Nevertheless, it is important to differentiate between controllable parameters that may be adjusted at run-time (e.g. frequency of re-building a network tree) from those requiring a coordinated network re-set to prevent malfunctions in the normal behavior (e.g. switching radio channel). The former are particularly interesting for defining dynamic rules allowing run-time trade-offs. The latter, on the other hand, need to be analyzed at design time to determine a *best* start-up configuration maintained for the entire system life-time[1].

Environmental parameters, such as the dimensions, profile, and meteorological conditions of the area of deployment must be taken into account when exploring the system set-up. These, and other parameters such as the mobility of nodes, network density, or exposure to external sources of interferences, will determine the set of values that controllable parameters can take.

As an example, defining two metrics as the *average latency of end-to-end message transmissions* and the *overall energy consumption of the entire network* allows trading

[1]Note that in most cases it is possible to run a distributed algorithm forcing the adoption of new setting for any given parameter. However, these methods produce a significant alteration of the normal network behavior for a significant period of time similar to that of shutting down the network and re-programing the configuration, which is –in essence– equivalent to re-starting the network. Hence, in the context of this thesis applying such a method does not account for as a *dynamic* adjustment.

off energy consumption against average timeliness at a global level. However, note that the application of these trade-offs may lead to unexpected results unless the metrics are precisely constructed. For instance, decreasing the *average* latency at a global level may have the opposite effect on a subset of nodes. Similarly, the energy consumption may be unevenly distributed among all sensor nodes after the application of the trade-off, producing higher consumption in particular nodes in spite of the intended tendency.

Finding the right metric and exploring the correlation between the variable being traded off and other side effects is also fundamental. It is important to account for the effects of these two metrics on the global system performance. For example, reducing the energy consumption may affect the transmission power leading to disconnected nodes, or similarly increasing the transmission power may affect the interference range of nodes.

IV.1.3 System Models

Abstract models and simulation engines are valuable tools that speed up the analysis o large systems. Choosing or building a proper set of models reflecting accurately the involved subsystems may increase the accuracy while exploring the impact of parameters. However, the elaboration of a reasonably accurate abstract model for analysis or simulation research cannot be done arbitrarily.

It is very likely that many details from the original system escape during the elaboration of models. Even though most of them could be incorporated to increase the model accuracy, the level of complexity is often not practical for the abstraction and analysis purposes. Choosing the right level of detail will have a significant influence in the quality of the results, and may allow a satisfactory calibration of the model as well as the validation of the obtained set-points. Too little detail may hide important effects on metrics, while too much may overload the analysis jeopardizing the abstraction process.

IV.1.4 Enforcing Timeliness

Generally, the larger amount of resources that are available to a sensor node, the higher the chances that a smart algorithm succeeds increasing the system performance. In particular, the more energy is available to the system –enabling e.g. higher transmission power, multiple message transmissions, larger duty-cycles– the more likely a method will succeed fulfilling the imposed timeliness requirements. However, how to enforce this gain in timeliness performance is a question that depends in great measure on the particular use of resources followed by the method.

The transformation of feasible timeliness objectives into the necessary QoS requirements does not suffice to ensuring the correct timely behavior of a WSN. It is also necessary to introduce enough mechanisms to enforce fulfilling these requirements without violating the fundamental system constraints. Many of the existing algorithms (e.g. some explored in II.3.1) perform notably in achieving significant results at a low resource cost. However, most of them fail to dynamically adapt their behavior achieving a fine match between the user required timeliness performance and that achievable by the system. In fact, the design of most methods focuses on providing a best-effort performance

at a constantly low resource utilization. However, given that the dynamic properties of WSNs introduce significant variability in the provided timeliness performance, it seems more appropriate aiming at an unbalanced use of resources driven by the importance of achieving timely communication at each moment.

One promising way to doing so in harmony with the unstable timely behavior of WSNs is to apply selected trade-offs at precise instants of time. QoS trade-offs allow defining relations between several metrics influencing the target QoS level to compensate for the effects of the continuous variability of WSNs. The analysis of these relations often show that effective timeliness trade-offs can be defined with a number of different metrics. However, those related to the amount of energy consumed due to activity level of physical devices (e.g. radio transmission, memory accesses, CPU computation) represent the clear predominant metrics in WSNs producing a large impact on the overall timeliness performance.

Although the relation between energy consumption and timeliness performance provides enough flexibility to establish significant trade-offs, a careful analysis must provide the selected QoS set-points at which the improvement has a larger impact. The following sections explore the definition of significant QoS trade-offs based on a generalized system analysis.

IV.2 Analysis of Local Trade-offs

Taking advantage of network trade-offs is not a trivial task that can be done without previously acquiring a deep knowledge of the target system. The particular parameters or *set-points* conforming satisfactory trade-offs for a given system may not be valid for a different one or may even produce undesirable results in different deployments. Hence, a careful analysis of the entire system must precede in order to determine which are the parameters and set-points producing a significant impact on the metrics of interest.

The level at which the trade-offs are first defined and later applied is another important aspect deserving consideration. Local trade-offs involving the hardware platform, firmware, and application levels are suitable to perform individual adjustments of the operational parameters of each node. These could include the Operating System (OS) functional configuration, application specific settings, or the use of specific hardware modes. Global trade-offs, on the other hand, require a coordinated strategy establishing system-wide set-points that may not necessarily produce a significant impact in one particular node. However, once applied to the whole system they produce a larger impact reflected in the overall QoS level of the entire network.

With disregard of the level of applicability, defining QoS trade-offs requires the selection of a number of significant evaluative metrics. Being timeliness the main focus of this thesis, it seems coherent to choose the overall energy consumption as a contrasting metric to trade off with. Alternative options which may also contribute to significant set-points include among others variants like the energy per useful bit or radio transmission power, as well as transmitting or receiving time, and expected data rate.

IV.2.1 Definition of Local Trade-offs

A simple definition of local QoS trade-offs may consider several operational modes for the sensor node based on local configuration parameters conforming optimal set-points. However, for the correct application of these mode-based trade-offs it is necessary that the software and/or hardware platforms allow run-time adjustments of the parameters defining these set-points. The provided support at the OS level is decisive handling run-time configuration and accessing the necessary hooks to the hardware and communication platform. For example, the Operating System Abstraction Layer (OSAL) introduced in section II.2 and evaluated in detail in appendix A is designed to facilitate these operations easing the exploitation of local and global trade-offs.

If the system does not allow run-time re-configuration of all parameters, statically-defined set-points can be applied at design-time, although the effectivity of such an approach may be reduced due to the dynamic variability of WSN. Exploring partial trade-offs consisting of only those parameters in a set-point allowing run-time modifications may perform better and compensate for the variability in the system conditions.

In any case, local trade-offs are notably easier to define in comparison with global level trade-offs. The main reason is that the cause and effect remain at the same visibility level (i.e. the sensor node). Hence, the applicability of trade-offs at this level can be defined as a set of local rules applied individually at each node.

The number of parameters that may be available to constitute trade-off set-points depends very much on the application and platform specifics. However, these would certainly include parameters concerning processor power modes, utilization of optional computationally-complex algorithms (e.g. pre-processing of sensor readings), as well as the frequency and duty cycle of tasks within the node. In case of using certain sensors to monitor environmental variables, it is possible that the accuracy or granularity of the sensed data can be adjusted adapting to the dynamic run-time needs. These configurations could also be exploited while exploring local trade-offs.

Some of these controllable parameters which are not dependent on a particular platform are discussed with more detail in the following sections.

IV.2.2 Duty Cycles and Software Modes

The main functionalities of a sensor node may be provided by a number of tasks[2] that may or may not run in a coordinated fashion. In fact, modern WSNs may allow multiple applications running on the same platform. For example, it is not surprising to conceive a scenario consisting of three core applications (e.g. monitoring, data processing, and system diagnostic) running on top of the same platform.

In a multi-threaded OS (e.g. OSAL), it is not too complex to define a number of software modes enlarging or shortening the activation frequency of each application task. Some of these modes may even disable selected tasks and conform a series of execution profiles with different processor demands. The activation of these modes at run-time

[2]In this context, task refers to a functional unit of work performed by the system (i.e. the sensor node), with independence of its particular implementation, concurrent execution model, or interaction with other entities.

may allow the OS to enter in larger inactivity periods while enabling the activation of lower power modes.

The trade-off –in this case between timeliness and energy consumption– consists of adjusting the periods of running tasks so that the OS can switch to energy saving profiles during larger idle times. The larger these periods are, the lowest the processor consumption results, but consequently a larger latency in the execution of the system tasks is expectable. Note, however, that this trade-off cannot be exercised arbitrarily and could result in unaccomplished application requirements. It is the responsibility of the application designer –with assistance of the field experts–, to explore and define valid operation points as well as evaluate their performance with respect to the evaluation metrics.

IV.2.3 Local Application of Trade-Offs

Local trade-offs present the advantage of not requiring coordination with external entities. The frequency of adaptation can be generally as high or low as required without introducing excessive overhead in the system. However, this does not mean that the application of local trade-offs can be arbitrarily organized. The integration between the local decider and the system properties must be carefully established and the cost of switching between valid configurations included in the decision process.

Applications themselves may provide different levels of flexibility with respect to their inherent temporal constraints. The exploitation of this flexibility determines the boundaries within the operability range –from optimum timeliness performance to minimum acceptable levels. These boundaries are fundamental to establishing adequate triggers for possible QoS trade-offs and may help identifying possible application modes with divergent timeliness requirements.

Two significant aspects that may present a certain correlation are the traffic generation patterns and the granularity of the sensed data. Applications running complex algorithms may have the opportunity of choosing among different levels of data processing. The impact of these decisions may reflect on the utilization of energy-expensive resources (e.g. CPU, memory) as well as in the amount of necessary transmitted data.

Example

Consider the following example of a temperature sensor in a refrigerated truck with three possible application modes as a simple scenario:

Mode 0 – In the simplest and less processing intense mode, the task in charge of determining the temperature behaves as a switch. If the temperature goes above a threshold the task reports *high temperature*, and analogously, it reports *low temperature* if the temperature goes below a lower threshold. However, while the temperature remains between both thresholds, no information at all is transmitted.

Mode 1 – In the second mode, the same task keeps a history of temperature readings and based on a linear regression estimation predicts the way in which tempera-

ture changes over time. If the prediction shows an increasing or decreasing trend that could violate the temperature regulations, the task calculates the necessary correcting factor which will be transmitted to the actuators in charge. This action continues periodically and the compensation is transmitted regularly forcing the temperature to converge towards the desired level.

Mode 2 – In the most demanding mode, the task does not only acquire and control the temperature, but it is also responsible for producing a detailed log of fluctuation of the temperature level. In this case, if the content of the refrigerated truck has been exposed to inadequate temperature levels the information processed by the task is crucial to determining the consequences. Hence, in addition to the control described in mode 1, periodic messages are transmitted to report the current temperature level.

An application with the above modes qualifies for a local trade-off analysis as described in this section. In particular, the selection of modes 0, 1 or 2 may be triggered at run-time in response to the importance of controlling the truck temperature at each given instant of time. It is presumable that certain goods may require a less strict control than others. For example, different kinds of foods may require different ranges of temperature, being some of them particularly sensitive to temperature fluctuations while others tolerate certain variability. Hence, trading off accuracy in the temperature control at selected times produces lower CPU and memory utilization as well as less transmitted messages, which altogether results in a lower energy consumption.

Another aspect that can influence the local energy consumption at a sensor node is the application of data aggregation techniques. For example, extending the above example, it is conceivable that more temperature sensors will be present in a refrigerated truck (e.g. [Andersson 10]). Hence, assuming a multi-hop topology, nodes may be able to append their readings while forwarding other messages. Note that appending may be as simple as adding bytes to a message body or as complex as processing the data into average values or applying other sort of algorithms.

In a similar direction, data compression may reduce significantly the size of transmitted messages, although it is generally expensive in terms of required processing power as well as memory utilization. Nevertheless, the balance between the energy spent in these algorithms and the reduction in transmitted bits is another option to take into account for local trade-offs.

IV.3 Analysis of Global Trade-offs

Network or system-wide trade-offs comprise global QoS parameters whose repercussion domain affects the quality performance of the entire system rather than individual nodes.

Defining network trade-offs requires an extensive system analysis that is often not affordable in real deployments. Hence, the utilization of abstract models is in this case particularly beneficial. However, characterizing an complete WSN with the right amount of detail in accurate abstract models is a complex process.

This chapter explores an adequate methodology to reduce the complexity and achieve

a satisfactory analysis of network trade-offs.

IV.3.1 Application Modes and Models

Section IV.2 introduced local application modes to exploit the flexibility of different application demands at selected instants of time. An analogy can be made with respect to the global network application and system-wide trade-offs. For example, the application of a surveillance WSN in secured premises may have different requirements depending on a predefined time schedule (e.g. monitor intrusion outside office hours vs passive maintenance tasks during working time). Taking advantage of such operational modes, the network can be configured using specific set-points optimizing the selected metrics for each case (e.g. latency during monitoring vs energy saving in maintenance mode).

The identification of these modes require a knowledge of many aspects regarding the sensor network. The application experts may provide valuable assistance identifying significant use-cases taking into account details of the running application(s) as well as domain specific properties. There is a close relation between these use-cases and the application modes that will drive the trade-off analysis.

Once the different behavioral use-cases are defined, the correspondent application modes can be sketched as depicted in figure IV.1. Note that in some cases, if their impact on the metrics of interest is reduced, several use-cases may converge into the same application mode.

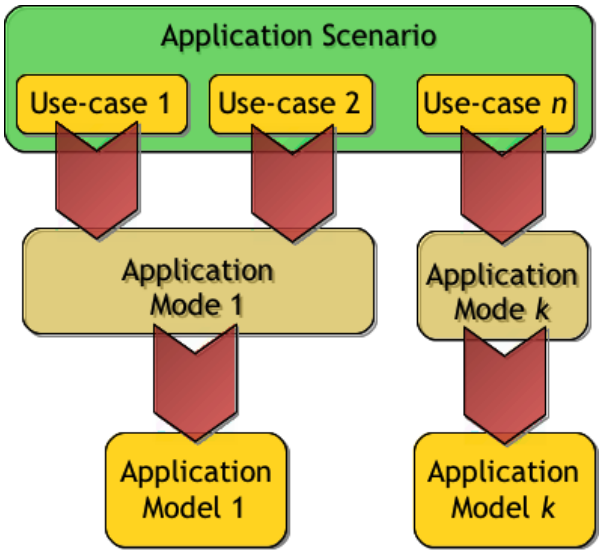


Figure IV.1: *Application Modes and Application Models*

Application modes can be directly converted into abstract models (e.g. simulation models) that will drive the trade-off analysis. During the process, it is important to consider which parameters need to be modeled. In particular, controllable and environment parameters that will conduct the definition of set-points.

Calibration of Models and Analysis of Trends

Using on abstract model to explore to a large extent possible network trade-offs has the advantage of not having to bother about most of the problems regarding real deployments. Moreover, it helps isolating the interactions from one abstraction layer to another (e.g. between different networking layers), which helps identifying potential trade-offs. However, it is difficult to validate the accuracy of specific set-points if the models are not validated against the real system.

The validation of models is a key step to properly defining effective trade-offs. Depending on the system complexity and application characteristics these set-points may or may not reflect the behavior of the system. A set of runs on a low-scale test-bed may help calibrating the models as well as identifying their accuracy [Rousselot 09].

Figure IV.2 shows an example of model calibration identifying the difference between experimental runs on a test-bed and simulation results. The figure depicts the measurements taken in two experiments, one via simulations and the other from a real-test bed. The metric of interest is the mean transmission latency of a short network experiment with respect to the radio transmission power. It is clear that although the tendency of both experiments show correlation, the validity of the simulation results is not accurate enough to define absolute values for each set-point.

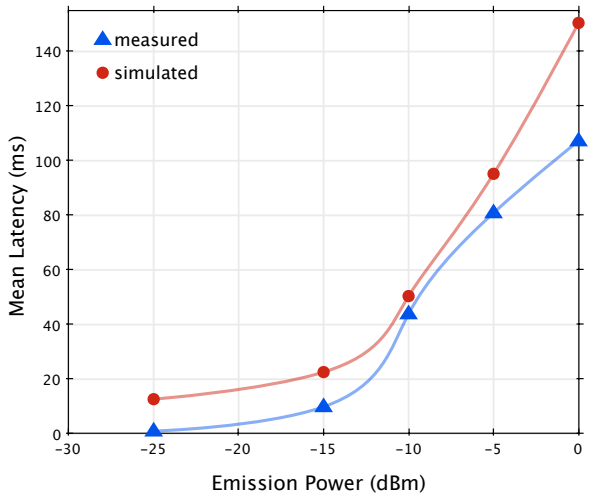


Figure IV.2: *Validation of Simulated Models and Identification of Trends*

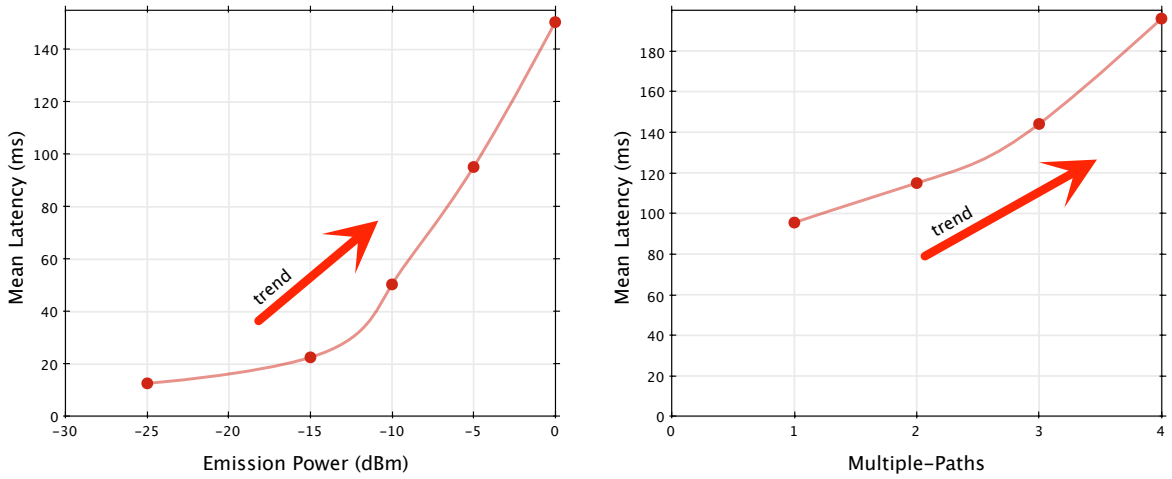
For simple scenarios governed by few parameters it is possible that the validation process shows a significant match between simulations and test-bed runs. However, for complex scenarios with multiple parameters it is not unexpected that the accumulated inaccuracies of abstract models lead to uncalibrated results.

For some of these cases, it may be sufficient to apply a correction factor to the set-points obtained via simulations. However, finding out these factors may require multiple calibration rounds with different scenario set-ups.

For other cases, the interrelation of unknown factors and hidden details from simulations and the real test-bed will prevent from applying a correction factor. However, the

tendencies or *trends* between both sets of set-points may suffice to provide significant information with respect to the correlation of metrics.

The analysis of trends consists on determining relations between metrics in spite of the absolute value for each simulated set-point. Validating these tendencies towards real test-beds is less influenced by the lack of detail and particularities of the real deployment. A series of experiments producing a small number of points in the axis of interest may suffice to extrapolate with satisfactory accuracy the correlation between metrics. For instance, an uncalibrated model may not be able to provide the exact values for the optimum transmission power, but may still provide a proportional notion of how much does latency decrease at each increment of the parameter.



(a) *Trends for the Emission Power* **(b)** *Trends for the Number of Multiple Paths*

Figure IV.3: *Analysis of trends for the impact of two parameters in the mean end-to-end latency.*

Once this correlation has been established, the direction of trends can be defined at different granularities. For instance, a simple model may determine that the correlation between metrics can be one of three possible: *positive*, *negative*, or *neutral*. Complex models may include additional relations such as *exponential*, *linear*, *cubic*, and others.

These trends are suitable for establishing trade-offs based on the relative impact of each parameter on the different metrics. Instead of forcing a fixed set-point, exercising trade-offs is performed by adjusting the relative value of the interesting parameters. In most cases, this trend-based trade-off models will lead to sub-optimal solutions unless feedback is collected while exercising the trade-off. However, collecting run-time feedback may be costly and not always efficient.

IV.3.2 Analysis of Set-Points

Figure IV.4 depicts the analysis of set-points that will lead to the definition of network trade-offs. Provided that the necessary abstract models have been created, the following

step is to determine the configuration space for the analysis of parameters. A complete analysis of the entire configuration space –either by simulation or analytically driven– may be unfeasible as the required computation effort expands rapidly with the number of parameters. Even for a small number of parameters, these can allow enough flexibility in the range of acceptable values that the available research efforts result overflowed. Further, a reduction of the configuration space is recommended in order to limit the magnitude of datasets to manageable sizes.

Several techniques may be used to limit the amount of values being tested for each parameter as well as shortening the analysis time. A particular one acknowledged to produce satisfactory results consists of first coarse distribution of values including the extremes with subsequent repetitions at a finer level around those particular areas showing interesting points. Nevertheless, this technique does not prevent from a pre-filtering of less interesting values based on the expectations and designer criteria.

At this point, each application mode is represented by one or several set-points. Whether these are absolute or indicative of a trend depends on the confidence gained with respect to the accuracy of the models during the calibration phase. Absolute set-points may be already collected as trade-offs, each of them producing a particular impact on the metrics. Trends provide relative information regarding the impact of one parameter on one or several metrics.

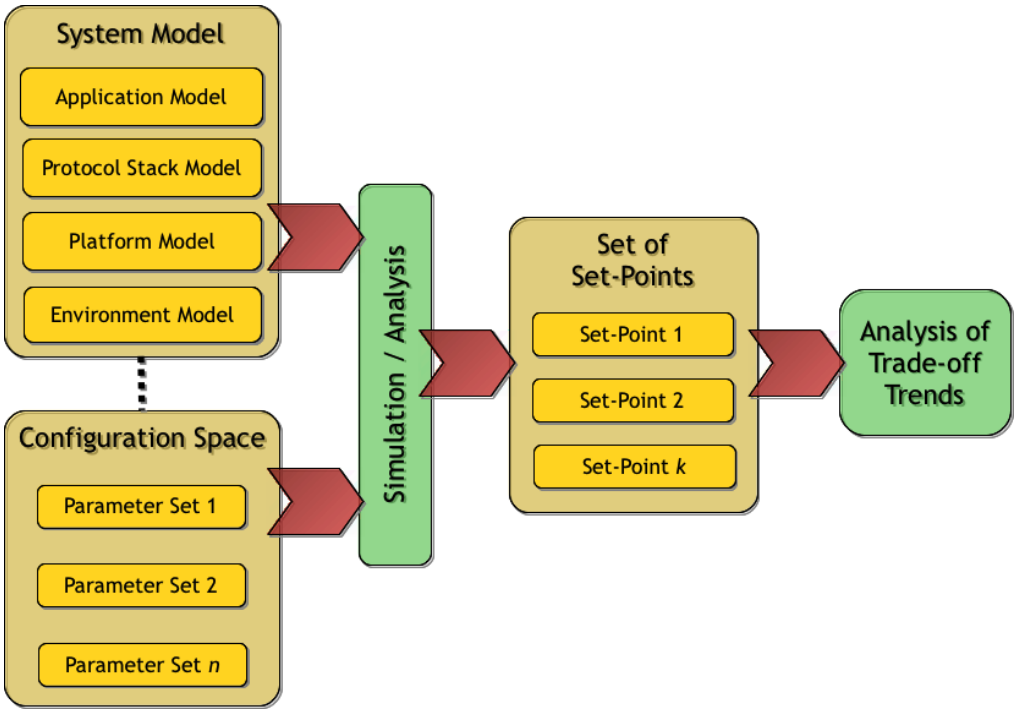


Figure IV.4: *Analysis of Set-Points*

IV.3.3 Application of Global Trade-Offs

Wether absolute set-points or ralative trends are defined, the correct application of global trade-offs may require additional mechanisms in order to guarantee consistency of the network configuration. In particular, certain global parameters may not allow at a given time different values for several nodes. For example, the waking up schedule of some Medium Access Control (MAC) protocols needs to be unanimous or the communication may not be possible.

A simple but effective solution to this problem consists of setting up those parameters offline. In doing so, the trade-off is reduced to finding an appropriate set-point that optimizes the metrics for the general case, although it is difficult to estimate if the effect will remain positive for the entire network life-time. Nevertheless, other parameters controllable at run-time may be available to compensate for the network dynamics and valid set-point with unconstrained parameters may still produce satisfactory results.

On the contrary, a distributed algorithm can guarantee that certain parameters are set uniformly across the network. For example, setting an instant of time to provoke the change while keeping a safe fall-back scenario if the communication is not successful after the change. Such examples may be necessary in the case of selecting different radio channel or extending the wake-up periods in MAC protocols. The algorithm must ensure that either all nodes switched to the new set-point or the configuration is reverted to the original situation.

IV.3.4 Monitoring of Global Metrics

Local trade-offs are relatively easy to apply and their consequences are immediately seen. However, another challenging issue with respect to global trade-offs is the observation of their effects at a system-wide level.

Although global metrics are properly defined to reflect these changes, their evaluation at run-time may be unfeasible or demand unreasonable amounts of resources. For example, monitoring the average energy consumption of a network may require each sensor node to report its actual consumption before and after the application of a global trade-off. However, by doing so the network traffic pattern is affected with the consequence of interfering the measured values.

It may be the case that some effects of applying network-wide trade-offs are visible at an individual node. For instance, noticing a change in the forwarding latency if the MAC duty cycle has been modified. However, the global effect of these changes may only be perceivable at a global level.

Generally, unconstrained nodes are the perfect entities to monitor these changes. In particular, sinks (also gateways or data-collectors) will be aware of global changes in the timeliness responsiveness and other time related variables. Other values concerning energy consumption metrics may require specific monitoring mechanisms or network statistic collectors, although either way these services will most likely be triggered by the sink itself.

IV.4 Chapter summary

This chapter extended the generalized notion of timeliness developed along chapter III with the exploration of timeliness QoS. The core of this chapter explored the definition and application of local and global QoS trade-offs and detailed a step-by-step methodology to analyze the most suitable system set-points.

Chapter V

A Timeliness Aware Routing Protocol

The application of the generalized notion of timeliness at run-time as well as the estimation of the end-to-end delay distribution require support from the network stack. Existing protocols can be easily adapted to express the timeliness performance by means of the generalized notion presented in chapter III. Similarly, the estimation of the end-to-end latency distribution can be embedded into routing protocols providing run-time estimations of path distribution delays.

This chapter presents an approach to a simple routing protocol consisting of a modified classic tree routing protocol, which generates paths based on the generalized timeliness notion. Each hop performs the local calculations as shown in section III.3, which are used to determine the best forwarding neighbor based on a routing tree. The sink periodically broadcast control messages to reconstruct the tree and collects information regarding the end-to-end delay distributions.

The purpose of this chapter is not to elaborate a complete protocol covering all important routing aspects, but rather to introduce the concepts elaborated throughout the thesis into a simple demonstrative protocol. Following the same ideas exposed along this chapter, the same adaptation may be done to more sophisticated protocols.

The content of this chapter is organized as follows:

Section V.1 overviews the basics of routing protocols based on rooted-trees.

Section V.2 describes the timeliness-aware routing protocol (TARP) embedding the estimation of the end-to-end latency distribution. The section details the amount of resources required by the protocol.

Section V.3 presents a simple extension to the protocol adapting it to multi-path and multi-sink scenarios.

Finally, the chapter concludes with a summary.

V.1 Introduction to Tree-Routing

A *rooted tree* is a *directed graph* in which one *vertex* (i.e. node) is designated the root and the connecting *edges* are directed towards (or away from) it. The relation between connected nodes tells wether they are *child* or *parents*. A child is connected to its parent which is closer to the root. A parent is a child itself with respect to the node that it connects in direction to the root. Similarly, a child may be a parent to other nodes. The exception to these relations are the root node, which does not have parent, and any node without children, which is called a leaf. Figure V.1 depicts these concepts graphically.

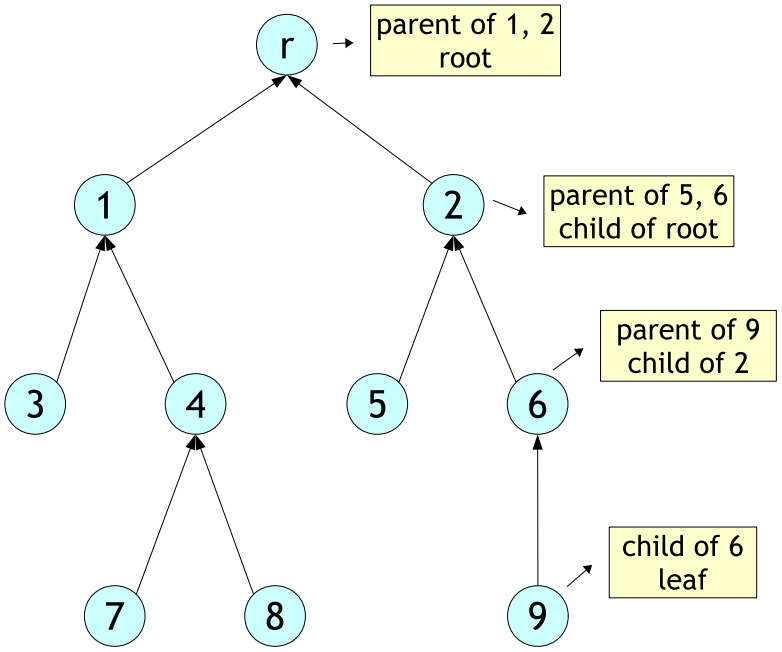


Figure V.1: *Example of a rooted tree.*

Rooted trees (*trees* from now on) are often used for the elaboration of routes in multi-hop networks. For each data collector (i.e. sink) a routing tree is maintained with the root set on the sink and the edges symbolizing the communication links between nodes. The criteria to choose among multiple parents are typically related to the quality of the link.

One of the advantages of such a routing protocol is that the logical tree can be distributed among the network components. Thus, none of the individual nodes knows the complete constitution of the tree, but only the reference to its parent. When communication is required, the source node forwards its messages to the known parent, which in response propagates them one level up –forwarding the messages to its own parent. Following the same principle, messages are forwarded hop by hop until reaching the sink (i.e. tree root).

Although it is a fairly simple mechanism, a routing protocol based on this technic has to consider a number of aspects, among others:

Tree building – The procedure to begin the construction of a network tree is normally triggered by the sink (or sinks). A common way of doing so is by broadcasting a sequenced message (e.g. *'HELLO'*). Upon reception, each child will annotate the source as its parent and proceed broadcasting a similar message with its own address on the headers. To prevent network flooding, the sequence is maintained so that the message is discarded if multiple receptions occur.

This process is typically repeated at a certain frequency to allowing new nodes to join the tree and the re-incorporation of disconnected nodes.

Validity of parents – As time passes, some of the links between nodes may become invalid (e.g. due to mobility or external factors). The protocol should contemplate a safe bound for the validity of the referred parents, which in most cases depends on the specific network set-up (e.g. existence of mobile nodes or mobile obstacles, exposure to nature phenomena, and others). After the expiration of this time-out, the parent is invalidated and a the tree must be re-constructed.

Active vs passive re-building – When a node is left without a parent –either due to the invalidation by time-out or because of not having received any of the broadcast messages– there are two alternative approaches to follow. On one hand, the node may become active and broadcast the pertinent control message forcing a re-construction of the network tree (e.g. `'BUILD-TREE REQUEST'`). On other hand, it may stay passive and wait for the sink to initiate the next periodic tree re-build round.

Selection of parent – Tree-building messages are generated at first by the sink and then forwarded by *each* of its children nodes. Hence, any given hop may receive several copies of the same message coming from different sources (e.g. one from each child, depending on the radio coverage). This is an effect of sharing a common medium with multiple paths from any given node to the sink.

There are multiple possible criteria to select which of the multiple received messages comes from the most suitable path. A common approach consists of choosing the parent as the forwarder of the first received copy, as it proved to lead the fastest path. Other strategies may delay the decision and relay on the highest received signal strength (RSSI) among all candidates. Alternatively, complex methods may keep a history of "best candidates" and base their decision not only on the current iteration but also on previous references.

V.2 Timeliness Aware Tree-Routing Protocol (TARP)

Based on a simple tree-routing protocol as described in section V.1, the following adaptations introduce the estimation of the end-to-end latency using the tree-building messages generated at the sink. The overhead in terms of communication is reduced to a few additional bytes appended to the regular tree-building messages. The memory requirement is limited to a pair of local variables for the local estimation of forwarding latency plus two additional variables for the storage of the path estimated latency.

V.2.1 Local Calculations

The computation of the local estimation of parameters for the distribution of forwarding latency is divided in the following steps:

(a) Upon transmission of a message, each hop calculates its forwarding latency as shown in equation III.6 (chapter III).

  (i) The routing layer receives notifications from the MAC layer whenever new messages or acknowledgements of reception (ACK) arrive.

  (ii) Locally, the timestamps of each transmitted message are stored until reception of an ACK.

  (iii) The node performs the estimation of its own forwarding latency as shown in section III.3 upon notification of an ACK.

(b) The results are stored in a set of local variables shown in table V.1, whose value is updated after a new message is forwarded.

Local Estimation Variables	
Variable	Type
Iteration (it)	(int)
Local $\bar{x_{it}}^*$	(int)
Local $s_{it}^{2\,*}$	(float)
Exponential factor (α)	[constant]

Table V.1: *Required local variables for the estimation algorithm.*

V.2.2 Initial Protocol Adaptations

The first adaptation to the tree-routing protocol embeds the estimation of the end-to-end latency in the process of building a routing tree. The description of this process is as follows:

(a) The procedure to establish a routing tree is started at the sink by periodically broadcasting `'BUILD-TREE'` messages with a sequence number and end-to-end estimation fields as shown in table V.2.

(b) Upon reception of one such message the receiver checks the sequence number and compares it with the stored variables:

  (i) if it is less than the last sequence seen, it means that a newer broadcast message arrived, hence this message is old and can be discarded.

  (ii) if it is greater than the last sequence seen, it stores the three parameters (see table V.3).

  (iii) if it is equal to the last sequence seen, the node evaluates the parameters with the local variables. Only if the new parameters show improvement with respect

to the old ones the table is updated. Otherwise the message is discarded.

(c) If the message has not been discarded, the source address is taken from the headers and annotated as the parent node. Following, the message is prepared to be re-broadcast:

(i) the header is updated to contain the address of the new sender,

(ii) the estimation variables are updated adding the local forwarding estimation of this node,

(iii) the sequence number is left untouched.

(d) The updated message is broadcast, and the process repeats until the distributed-tree is formed.

Following this procedure, a `'BUILD-TREE'` message carries the end-to-end distribution parameters from the sink to the each other node on the tree. Hence, nodes get an estimation of the end-to-end delay distribution to that particular sink. This information is made available to the application layer to contrast the timeliness requirements against real run-time estimations.

Header	Payload		
—	Sequence Number (int)	Estimated $\bar{x}^*_{it}$ (int)	Estimated $s^{2^*}_{it}$ (float)

Table V.2: *Format of a `'BUILD-TREE'` message.*

The protocol does not rely on the latency of a single message, nor static metrics like the distance to the sink, for the selection of the parent nodes. Instead, it generates routes based on actual timeliness performance calculated at run-time.

Protocol Variables	
Variable	Type
Last Seen Seq. Num.	(int)
Path $\bar{x}^*$	(int)
Path s^{2^*}	(float)
Parent node	(address)

Table V.3: *Required local variables for the tree-routing protocol.*

V.2.3 Evaluation of Best Path

During the distributed formation of a routing tree, it may be necessary to evaluate several routing paths with different estimated parameters ($\bar{x}^*$, s^{2^*}). There are two

alternative variants for the selection of the best candidate: on one hand, the path showing the shorter average delay ($\bar{x}^*$), which proves to be faster on *average*; on the other hand, the path with shorter variance (s^{2^*}), which proves to be more stable.

The presented protocol takes the decision based on a combination of both parameters. The intended result is to select the path with the shorter expected end-to-end delay. Looking at the Normal distribution (see figure III.5), nearly 68% of the values are expected to fit within the interval ($-\sigma$, $+\sigma$). Hence comparing the upper limit of this interval (i.e. $\bar{x}^* + s^*$) from both paths, the one closer to 0 is preferred. Figure V.2 depicts the area of interest.

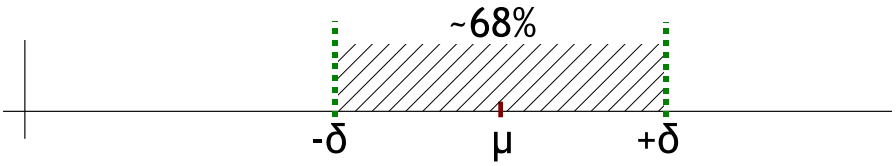


Figure V.2: *Limits of the interval with nearly 68% of points in a Normal distribution* $N(\mu, \sigma^2)$.

Note that for the simplicity of the calculation the comparison is done between $\bar{x}^* + s^{2^*}$. Given that the calculation is done to find out which of the intervals is closer to 0, the difference between using s^{2^*} or s^* does not affect the result.

V.3 Multi-Path Extension

The original design of the timeliness-aware protocol assumed a network with one single sink. Hence, a single record for the information regarding the forwarding node (i.e. parent) and the estimated variables suffice. As an extension to the protocol, this section describes a multi-path and multi-sink extension.

The extension handles multiple paths to the same sink and multiple sinks indistinctly. The information for each one of them is stored in a separate data-record stored in local memory as shown in table V.4. The maximum amount of records that each node can store depends on a configurable setting that determines the memory utilization of the protocol.

When a node receives a `'BUILD-TREE'` message from any of the sinks, it identifies the immediate source (i.e. last forwarder) as well as the original source (i.e. sink that originated the message). With these two parameters, it searches in the table for any matching record.

The procedure depends on the result of this search with respect to the pair *(sink, parent)*:

1. *(match, match)*: if both fields match, then the data record is updated as it is the same path with updated values,

Protocol Data Record	
Variable	Type
Sink	(address)
Parent node	(address)
Last Seen Seq. Num.	(int)
Path $\bar{x}^*$	(int)
Path s^{2^*}	(float)
Age	(timestamp)

Table V.4: *Variable record for the extension to multi-path and multi-sink.*

2. *(match, no match)*: a new path for one of the sinks is found. If there is no free position in the table, the record is updated with the new path. Otherwise a new record is inserted,
3. *(no match, match)*: an already existing parent is part of a new path to another sink. In this case, a new record is inserted. The data of the old record cannot be updated as it refers to a different end-to-end path,
4. *(no match, no match)*: if none of the records matches any of the indexes, a new record is entered in an empty entry or replacing an old entry if none is available.

If a new entry has to be inserted and there is none available, the protocol selects the oldest entry in the table to be replaced. This is done by checking the *age* field, which is updated after insertion and every time that the record is updated.

Selecting the appropriate node to forward messages to is done by choosing the entry in the table with the parameters better adjusting to the traffic criteria. The selection of an an appropriate parent connecting to the addressed sink requires a simple search on the table. In the case of multiple entries, the protocol my decide which one matches the intended performance. For instance, a node sending both *urgent* and *regular* messages may select different paths for each kind based on the estimated parameters.

The limitation on the number of multiple paths and sinks that this method can handle is bounded by the amount of memory used by the protocol. The more available memory the larger amount of entries may be stored. However, a large number of entries may introduce higher overhead in handling the table. This extension has been tested for small tables of 3 to 5 entries for which the overhead was practically negligible.

V.4 Chapter summary

This chapter introduced a timeliness aware routing protocol based on the concepts already introduced along this thesis. This protocol provides continuous estimations of the end-to-end timeliness performance and made them available the applications layer. The latency information allows the application determining whether the end-to-end requirements can be achieved with the current network conditions and enabling the exploitation of trade-offs if necessary.

Chapter VI

Evaluation

This chapter presents an evaluation of the main concepts introduced along the thesis. A number of scenarios evaluating different aspects are described and implemented. The evaluation is carried out by means of both simulation tools and small-scale test-bed experiments. The core of this chapter focuses on the evaluation of the routing protocol described in chapter V and provides hints for the application of the trade-off concepts presented in chapter IV.

The chapter is organized as follows:

Section VI.1 overviews the evaluation environments and the scenarios that serve as a basis for the evaluation process. These scenarios are conceived taking into account properties extracted from real deployments as well as small set-ups to validate specific network aspects. The section introduces the real test-bed platform as well as the simulation environment in which the evaluation experiments are conducted. A description of both platforms, including implementation and configuration details is also presented.

Section VI.2 summarizes the results obtained during the evaluation process. The section is divided into two parts, corresponding to the results obtained by simulation runs, and the correspondent figures from a number of controlled test-bed experiments.

Section VI.3 concludes the evaluation chapter with a discussion regarding the presented results. The results and figures presented in section VI.2.2 are commented and compared with the simulations in section VI.2.1.

Finally, the chapter concludes with a summary.

VI.1 Evaluation Environments

This chapter elaborates a detailed evaluation of the main concepts presented in this thesis. The evaluation is driven by two complementary methods, based respectively on the use of simulation tools as well as a number of small-scale test-beds carried out in a lab deployment. This section describes both evaluation environments as well as the scenarios in which the experiments are conducted for both cases.

VI.1.1 Description of Simulation Environment and Tools

The definition of simulation models used on the evaluation of the work presented in this thesis is based on the *Omnet++* framework [Varga 01], [Community 10] and related tools. OMNeT++ is a modular simulation framework –free for academic and non-profit use– written in C++ and specifically designed for network simulations. The modular architecture allows the development of independent models supporting domain-specific functionalities.

Among these modular frameworks, the *Mobility Framework* (MF) [Koepke 10] is intended to support wireless and mobile simulations within the OMNeT++ environment. The framework provides support for node mobility, dynamic connection management and wireless channel models.

Omnet++ versions 3.3 to 4.1 and the Mobility Framework version 2.0[1] constitute the core of the simulation environment in which the evaluation of this thesis is based.

The evaluation of simulations results is done by comparing the measured end-to-end delay of messages with the estimated distribution. The timeliness perception at the application (μ, σ^2) is recorded into the transmitted messages. Upon reception of each message at the sink, the final end-to-end delay experienced by the message is normalized with respect to these values. This allows the comparisons of streams of messages with different estimated parameters, which may change continuously due to the network activity.

VI.1.2 Description of Test-Bed Platform and Lab Deployment

The set-up for the evaluation on small-scale test-beds is motivated by the EU Framework 6 IST Project "Wirelessly Accessible Sensor Populations" (WASP) [Consortium 10a]. The project dedicated a significant number of resources to the implementation of software and firmware modules for the BSN hardware platform [ICL 10]. The lab-experiments carried out for the evaluation of this thesis are based on this platform and part of the modules implemented within WASP.

The main characteristics and components of BSN nodes[2] are listed following:

- board size $19mm \times 30mm$
- TI MSP430 [msp 04] 16-bit ultra low power RISC processor

[1] Part of the work presented in this thesis was originated over MF v.2.0 for Omnet++ 3.x, and later ported to MF 2.04 for Omnet++ 4.x.

[2] The test-bed platform described in this section corresponds to the BSN node v.3.

- 48KB flash/ 10KB RAM
- 8 channels 12-bits ADC
- 2 channels DAC
- 2 USART
- TI CC2420 [cc2 08] radio transceiver
- Communication range of ca. 50m (indoors) and ca. 125m (outdoors)
- Fitted with a miniaturized chip antenna
- 4MB external EEPROM

The BSN platform is distributed in so called *development kits*. Each kit consists of one USB programming board, one sensor board, one battery board, one prototype board and a pair of BSN nodes.

The WASP tool-chain [Consortium 10b], developed within WASP as a compilation of tools and utilities for the development of software for Wireless Sensor Networks (WSN), provides sufficient resources for developing, testing and debugging the implemented software modules. The Operating System Abstraction Layer (OSAL), implemented on top of MANTIS OS [Bhatti 05] as described in detail in appendix A, provided the link between the software and hardware platforms.

VI.1.3 Description of Network Stack Protocols

As far as possible, the repertory of protocols in charge of the communication in both evaluation environments was chosen to be equivalent. The WASP project provided a valuable set of protocols and simulation models that facilitated this goal without having to compromise the objectives of this work.

The network stack deployed in the test-bed experiments coincides partially with the *HC-stack*[3]. Omnet++ models for the main components of this stack are also available within the project consortium for the evaluation based on simulations. These validated models are calibrated to reflect with accuracy the behavior of the original protocols.

A brief description of the protocols constituting the network stack follows:

CC2420 – This IEEE 802.15.4 compliant RF transceiver [cc2 08] is documented in chapter II. The transceiver is part of the BSN node architecture deployed in the evaluation experiments. The simulations included a validated Omnet++ model reproducing accurately the behavior of this Radio Frequency (RF) chip [Rousselot 09].

WiseMAC – WiseMAC [El-Hoiydi 04] is an energy-efficient MAC protocol based on synchronized preamble sampling specially designed for WSNs. In WiseMAC, all sensor nodes independently sample the medium at a constant period (Tw). If the medium is busy, the node listens until a data frame is received or the medium becomes free. To guarantee the reception of messages, a wake-up preamble of size Tw is transmitted in front of every message. After the reception of a message, a

[3]Internally labeled as "HC-Stack" for being applied on the *herd-control* scenario [Lokhorst 07].

node will try to synchronize its wake-up interval to that of the transmitter, hence waking up short before the transmission of a new frame.

The evaluation documented in this chapter is carried out with an Omnet++ model and a software implementation of WiseMAC. In both cases, the configuration of parameters was chosen to present an equivalent behavior.

FTSP – [Maróti 04] describes the Flooding Time Synchronization Protocol (FTSP), which uses low communication bandwidth providing a robust synchronization mechanism. FTSP produces periodic flooding of synchronization messages, and implicit dynamic topology update utilizing MAC-layer time-stamping and error compensation including clock skew estimation.

TARP – Chapter V introduced the timeliness aware routing protocol designed to evaluate the applicability of concepts presented in this thesis. A model of this protocol is used for the evaluation in the simulation environment in Oment++. For the test-bed experiments, parts of the protocol were introduced in the test-bed implementation validating the scenarios.

VI.1.4 Description of Scenarios

The evaluation carried out in this chapter is based on two main scenarios evaluated under a diversity of parameter configurations. Both scenarios are implemented in the two complementary evaluation environments and the results are presented in section VI.2.

Scenario 1: Linear Network

The first simple evaluation scenario consists of a linear network of i hops as depicted in figure VI.1.

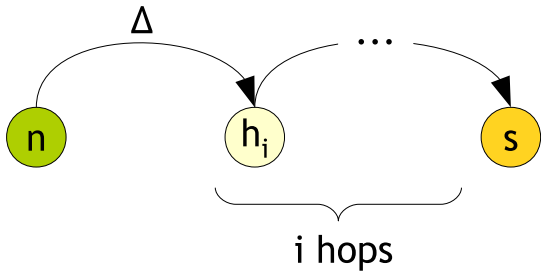


Figure VI.1: *Evaluation scenario 1: Linear network composed of one sender node (n) generating periodic traffic at rate Δ and one sink (s) separated by a number of hops ($h_i, 1 \leq i \leq 4$).*

In this set-up, the routing path is manually fixed (i.e. static) and only the traffic generated by the sender node (n) towards the sink (s) is present. The main purpose of such an scenario is to validate the end-to-end distribution algorithm presented in chapter III exploiting the generalized timeliness notion, as well as to demonstrate the

validity of the assumptions with respect to the distribution of the measured end-to-end latency.

Scenario 2: Routing Path with Cross-Traffic

The second scenario is a variant of the previous with the addition of cross-traffic. Figure VI.2 depicts the scenario.

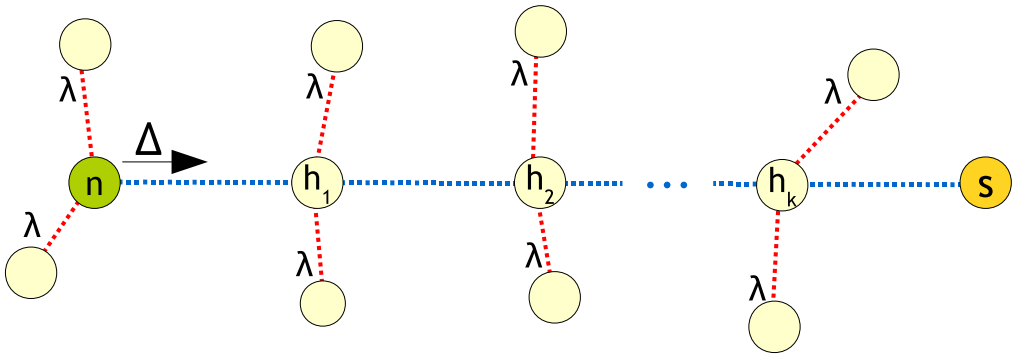


Figure VI.2: *Evaluation scenario 2: Routing path of length k with nodes logically connected to two neighbors generating cross-traffic with exponential time between packets λ. Sender node (n) transmitting periodic traffic to sink (s) at rate Δ.*

The routing path in this scenario follows the logical hierarchy depicted in figure VI.2. Hence, each hop of the analyzed route has to forward cross-traffic of two additional neighbors as well as its own. These messages are forwarded towards the sink (s) along with control messages evaluating the end-to-end latency (i.e. those generated in n). The traffic generated at each neighbor node follows an exponential distribution simulating cross-traffic coming from multiple sources, while n generates messages periodically.

This topology is representative of the section of a WSN with multiple arbitrary traffic sources. Note that the logical connection between hops (i.e. parents and children) does not limit the interference range and potential collisions between physically nearby nodes. Hence, as in a real deployment, ongoing parallel transmissions may be affected between neighboring nodes.

The configuration of this scenario considered three parameters, namely: path length ($|rp|$); period of messages generated in n (T); and, mean inter-arrival time for cross-traffic generated in the neighbor nodes (λ). The following list shows the range of parameters considered for the evaluation of this scenario:

- path length: $|rp| = \{5, 10\}$,
- n generating periodic messages transmitted to s with period $T = 30s$,
- messages aggregate the estimated parameters at each intermediate link (Equation III.6).
- $\alpha = 0.9$

- each hop in the path has two neighbors simulating cross traffic following a Poisson distribution with parameter $\lambda = \{30s, 60s, 120s, 480s, 1200s\}$,
- distance between nodes following a uniform distribution with range 8 to 20 meters.

VI.2 Evaluation Results

This section presents the most significative results obtained in multiple experiments performed according to the previously described evaluation methods. As the number of figures and results is large enough to overload this chapter, some of them have been moved to appendix B together with additional examples complementing the evaluation.

VI.2.1 Evaluation of Simulated Scenarios

With the aid of simulation tools, a large number of scenarios with multiple combinations of parameters and conditions were tested. The following are some of the results that validate the use of the generalized timeliness notion and the timeliness aware routing protocol.

Simulations Scenario 1

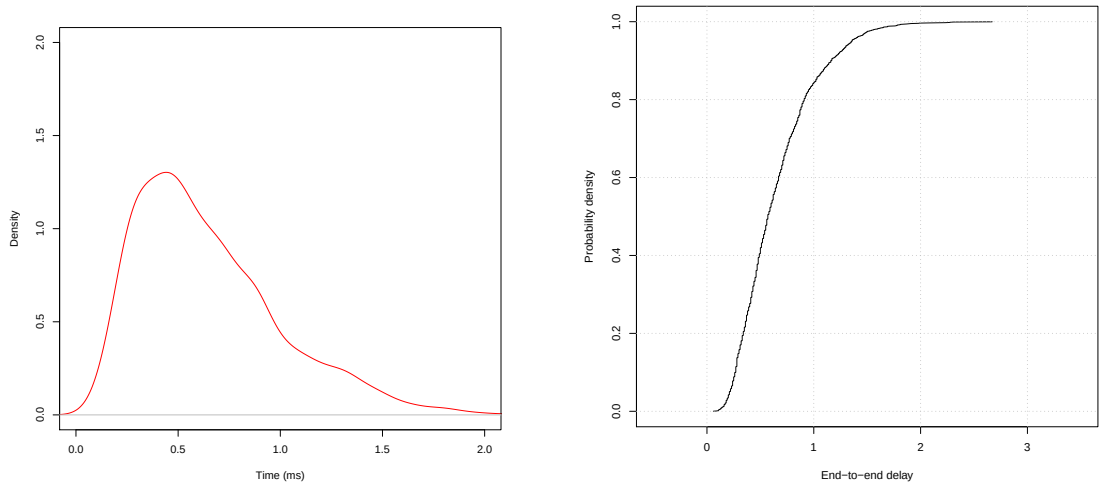
Parameter	Value
WiseMAC Sample Period	$250ms$
Transmitted Messages	100000
Time Between Messages	$30s$
Topology	Scenario 1

Table VI.1: *Summary of set-up for the simulation experiment as described in scenario 1.*

Following the topology described in scenario 1 and the set-up summarized in table VI.1, figure VI.3 shows the probability density function (PDF) obtained by multiple simulation runs.

Figure VI.3b depicts the estimated cumulative distribution function of the same experiment. Both figures, illustrate the relation between the bounds of the time interval and the achieved probability. Note that both factors are directly dependent on each other.

With the availability of this information at run-time, an application is able to estimate the end-to-end latency based on the current network status. In the scenario depicted in these figures, nearly 80% of the messages arrive at the sink in less than 1 second. This information is already a valuable indicator for the end-to-end latency. However, note that the distribution estimation is dynamically changing at run-time, adapting to the evolution of the network, the environment, and traffic load.



(a) *PDF of the end-to-end latency distribution* **(b)** *CDF of the end-to-end latency distribution*

Figure VI.3: *Distribution of end-to-end latency in a simulation experiment following the topology of scenario 1 (PDF and CDF).*

The estimation provides additional information that can be exploited in the execution of network-wide trade-offs. As detailed in chapter III, the interval $(\bar{x}-s, \bar{x}+s)$ contains approximately 68% of the values. Hence, observing the evolution of $\bar{x}$ and s^2 before and after the application of network trade-offs it is possible to evaluate the real effects of the applied set-points, validating *at run-time* the parameter values obtained with the use of abstract models (e.g. as introduced in chapter IV).

Appendix B includes an extended example based on this scenario performing a comparison between the generalized and classic timeliness notions.

Simulation of Scenario 2

Table VI.2 summarizes the set-up for multiple experiments performed according to the description of scenario 2. This experiments show the relation between cross-traffic and the distribution of the end-to-end latency. The set-up includes situations with low traffic as well as examples approaching the point of saturation.

For convenience, the figures are grouped according to the cross-traffic parameter. A compact view of these results can be found in appendix B.

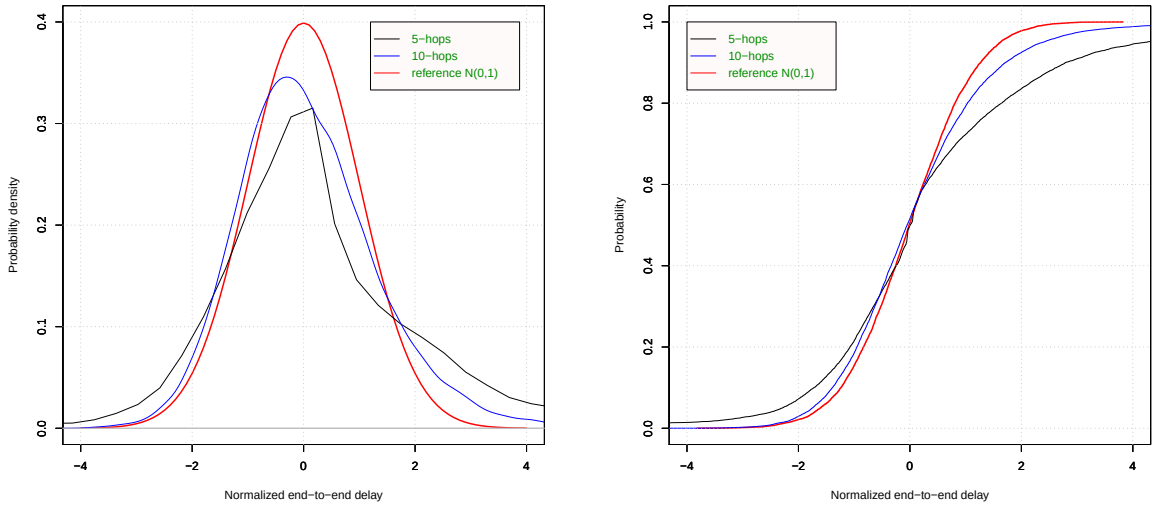
The analysis of each case shows interesting behaviors of the estimation algorithm. Figure VI.4, depicting the results for simulations with low cross-traffic ($\lambda = 1200s$) is significantly different depending on the path length. The accuracy of the curve showing the end-to-end estimation in the scenario with larger path length ($|rp = 10|$ is notably better than that of the shorter path ($|rp = 5|$).

On one hand, the larger path increases the number of links contributing with their local estimations, which –as discussed in chapter III– produces better estimations based

Parameter	Values
WiseMAC Sample Period	$250ms$
Transmitted Messages	100000
Period of Control Messages	$30s$
Parameter for Cross-Traffic	$\{30s, 60s, 120s, 480s, 1200s\}$
Path length	$\{5 - hops, 10 - hops\}$
Topology	Scenario 2

Table VI.2: *Summary of set-up for the simulation experiment as described in scenario 2.*

on the Central Limit Theorem (CLT). On the other hand, note that as the number of hops increase the number of neighbors in the scenario is proportionally larger, and therefore the generated cross-traffic increases influencing the estimation. In this case, the impact of more traffic is reflected in a more accurate estimation due to the higher frequency at which the measurements are updated.



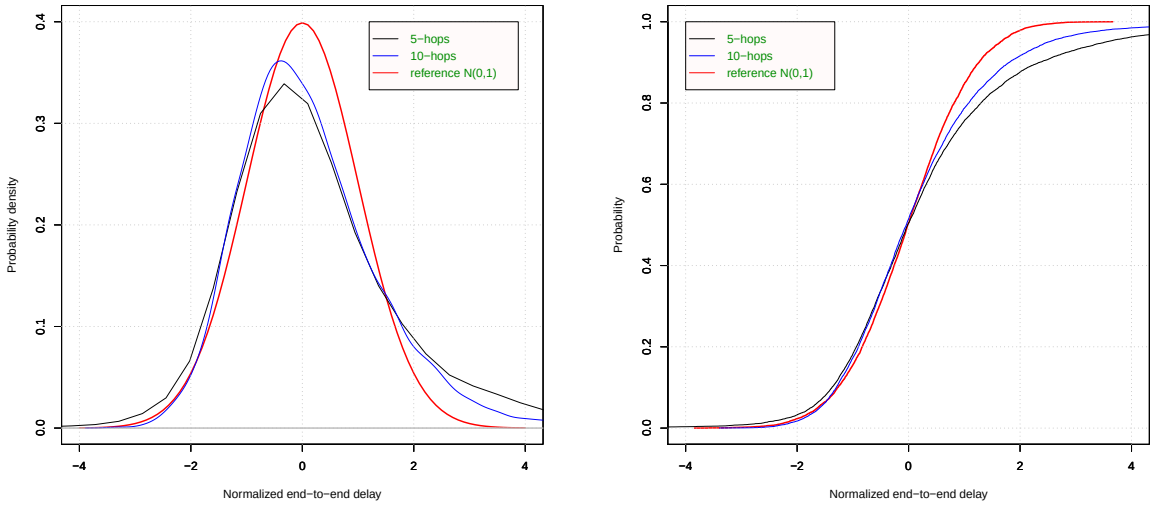
(a) *End-to-end distribution (PDF)* **(b)** *End-to-end distribution (CDF)*

Figure VI.4: *Estimated PDF and CDF corresponding to scenario 2 with* $\lambda = 1200s$ *and* $|rp| = 5$ *and* 10.

Figures VI.5 and VI.6 show two cases with a medium traffic load. Both estimations are notably accurate with respect to the reference distribution $N(0, 1)$. In both cases, the traffic is high enough to update sufficiently often the link forwarding estimation at each hop, contributing to the correct estimation of the end-to-end distribution.

The small error with respect to the reference curve $(N(0, 1))$ is due to the tail effect already mentioned in chapter III. The presence of a physical limit for the fastest possible transmission time introduces a minimum bound on the left side of the curve which is not

present on the other side. Hence, the deviation on the transmission time of a message is not equal in both sides of the central average value, introducing the asymmetry responsible for this tail.

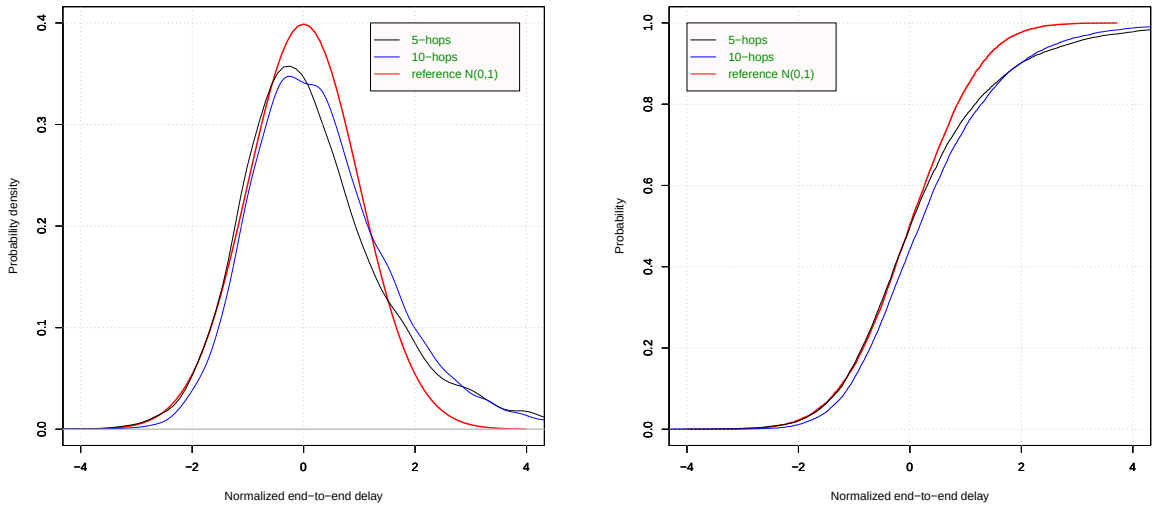


(a) *End-to-end distribution (PDF)* **(b)** *End-to-end distribution (CDF)*

Figure VI.5: *Estimated PDF and CDF corresponding to scenario 2 with* $\lambda = 480s$ *and* $|rp| = 5$ *and* 10.

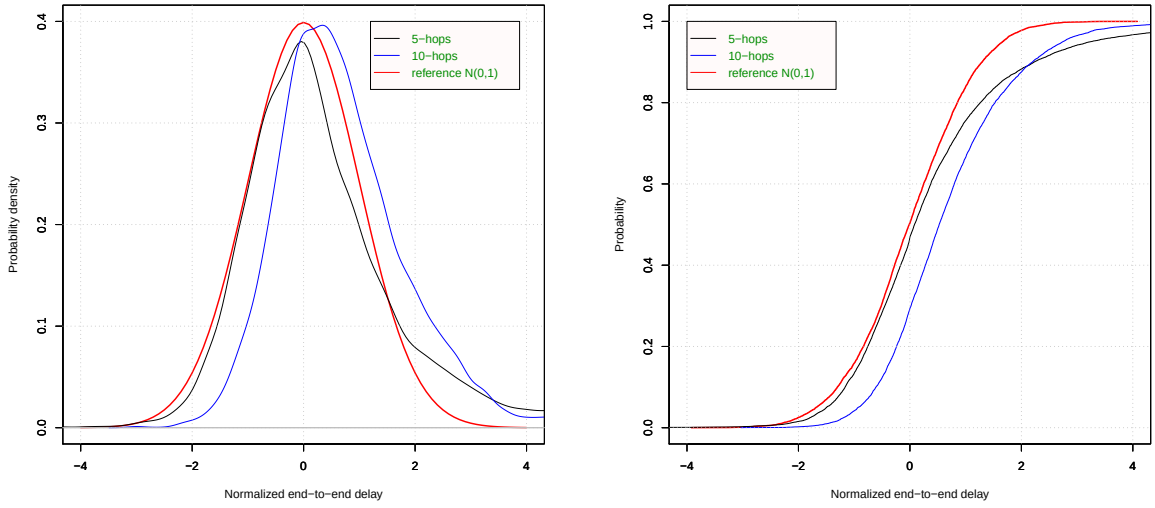
Figures VI.7 and VI.8 represent the other extreme with respect to the traffic load. In both cases, the amount of traffic approaches the network capacity, degrading the accuracy of the estimated distribution. In these cases, the number of hops contributed in a negative way as observed in figure VI.7. One of the reasons for this effect is the same increase in the network load for larger paths mentioned before, which explains that the result for $|rp = 5|$ still produces an acceptable level of accuracy.

Another reason for this decrease in the obtained accuracy for scenarios close to network saturation lies in the higher amount of missed acknowledgments. When an acknowledgment is missed, the forwarder node considers that the message was not received, and hence proceeds with its retransmission. However, during these experiments it was detected that in many cases the message was properly delivered and the receiver continued forwarding it further away. The result is that the calculated latency of the message at the sender node is notably worse than the real delay experienced by the message. Such phenomena are expected to happen in WSNs, and this result shows that measures must be taken to counterbalance their effects in networks with high traffic loads.



(a) *End-to-end distribution (PDF)* **(b)** *End-to-end distribution (CDF)*

Figure VI.6: *Estimated PDF and CDF corresponding to scenario 2 with* $\lambda = 120s$ *and* $|rp| = 5$ *and* 10.

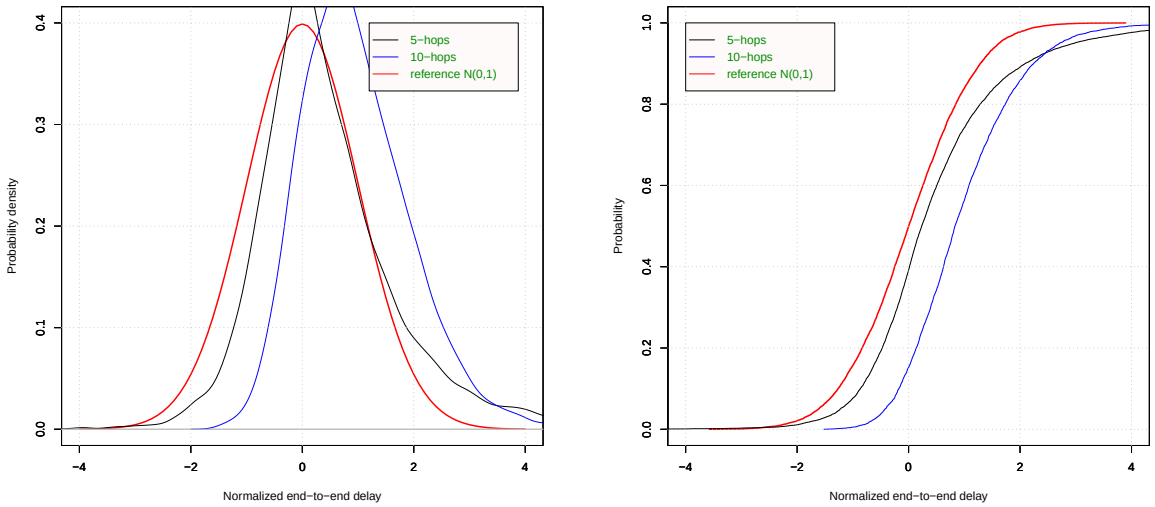


(a) *End-to-end distribution (PDF)* **(b)** *End-to-end distribution (CDF)*

Figure VI.7: *Estimated PDF and CDF corresponding to scenario 2 with* $\lambda = 60s$ *and* $|rp| = 5$ *and* 10.

VI.2.2 Test-Bed Evaluation

For the validation of the previous simulation work, a number of test-bed runs were carried out. Due to physical and practical limitations, these runs were divided into a



(a) *End-to-end distribution (PDF)* **(b)** *End-to-end distribution (CDF)*

Figure VI.8: *Estimated PDF and CDF corresponding to scenario 2 with* $\lambda = 30s$ *and* $|rp| = 5$ *and* 10.

Parameter	Test-Bed 1	Test-Bed 2
WiseMAC Sample Period	$250ms$	$250ms$
Transmitted Messages	1000	$500ms$
Time Between Messages	8s	10s
Topology	Scenario 1: 2 hops	Scenario 1: 3 hops

Table VI.3: *Summary of the set-up for the test-bed experiment*

number of small lab experiments with the purpose of analyzing the latency of messages and its distribution patterns.

End-to-end Latency

Figure VI.9 shows the end-to-end latency for a test-bed run with 1000 messages transmitted in a 2-hop network following the topology depicted in scenario 1. A different 500-messages run in a 3-hop network is plotted in figure VI.10. Table VI.3 summarizes the test-bed set-ups for both experiments.

Note that despite the initial peaks, both figures show a similar behavior. Even though both networks are small, the figures show a significant variability in the achieved end-to-end delay. As expected, for short experiments with low traffic levels, the latency appears to be approximately centered around a constant average transmission time and the variability behaves as if it were bounded.

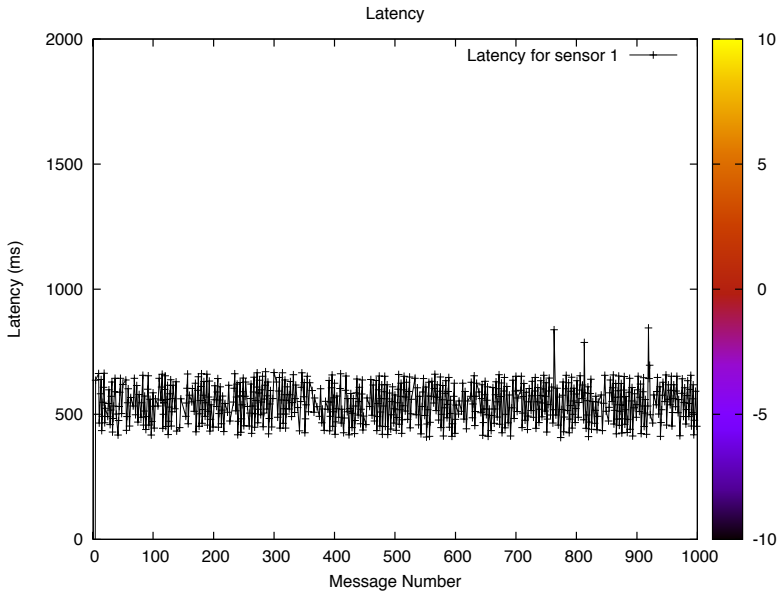


Figure VI.9: *End-to-end latency for a 1000 messages test-bed run in a 2-hop network as in scenario 1*

Nevertheless, even though the conditions of these experiments were favorable for a successful result, the number of packet losses is relatively high. Figures VI.11 and VI.12 show the packet loss curve with respect to the transmission of messages. Note that the distribution of packet losses is approximately constant (i.e. increasing linearly) and it is not caused by a temporal anomaly affecting the transmission.

This large number of packet losses is not unusual in WSNs and most of them are due to radio irregularities, collisions, or transmission errors. This motivated the inclusion of a simple algorithm taking account for the index of packet losses in TARP. The protocol performs a selection of the most suitable path for the timely transmission of messages evaluating the packet-loss index together with the end-to-end estimations. Note that any deployment exposed to realistic scenarios will suffer from this same issue, which in the experiments performed during this analysis was significantly stressed if mobile nodes or weak links were present.

One of the lessons that can be extracted from this initial evaluation is that the application of network trade-offs should take into account the packet loss index (or else, the delivery ratio) and perform run-time adjustments based on the observed network behavior. The large variability observed in this metric during additional runs of these set-ups –in practically identical conditions– suggests that the performance in larger deployments can be severely affected if only the metrics of interest (e.g. timeliness, energy consumption) are taken into account. In some cases, it was observed that a selection

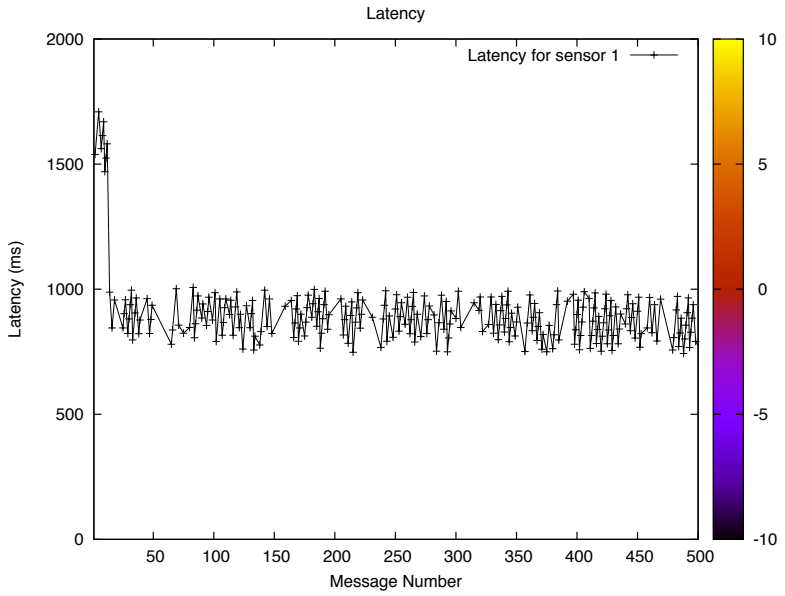


Figure VI.10: *End-to-end latency for a 500 messages test-bed run in a 3-hop network as in scenario 1*

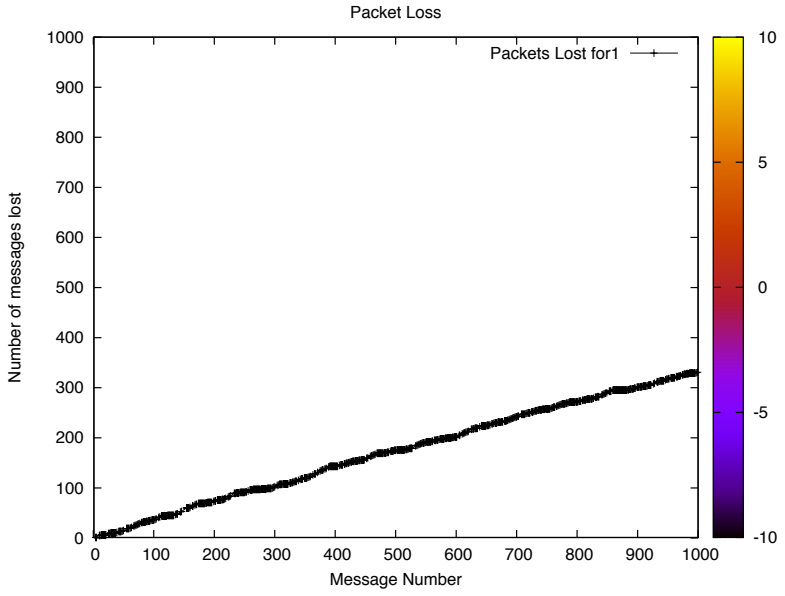


Figure VI.11: *Packet loss for a 1000 messages test-bed run in scenario 1*

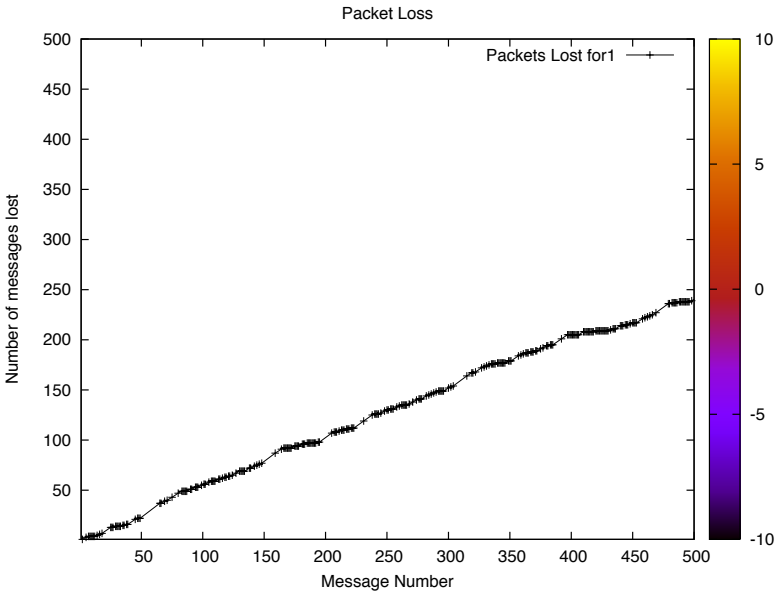


Figure VI.12: *Packet loss for a 500 messages test-bed run in scenario 1*

based on the fastest path resulted in a large amount of messages being lost. However, as the Quality of Service (QoS) metrics of interest did not reflect the delivery ratio, this was not initially detected as a bad choice.

Distribution of End-to-End Latency

The following figures show the evaluation of the end-to-end latency distribution of 1000 messages transmitted through a network of 2, 3, and 4 hops following the topology of scenario 1. As before, the network stack is composed of WiseMAC at the MAC layer, FTSP as time synchronization layer and simplified version of TARP as routing protocol.

In all set-ups, nodes were placed at selected relative distances trying to minimize the impact of the interference range. However, being a real test-bed, many uncontrollable radio anomalies may have been registered. Nevertheless, after a considerable amount of time (e.g. 1000 messages) the condition appeared stable and significantly representative.

For the experiments, the nodes were connected via USB ports to regular desktop computers running GNU/LINUX. The connection allowed debugging messages and traces to be displayed on the screen for further analysis.

For the sake of simplicity, the following figures show only the cumulative distribution function for the experiments in test-beds of 3 and 4 hops. However, appendix B includes the missing figures in section B.3.

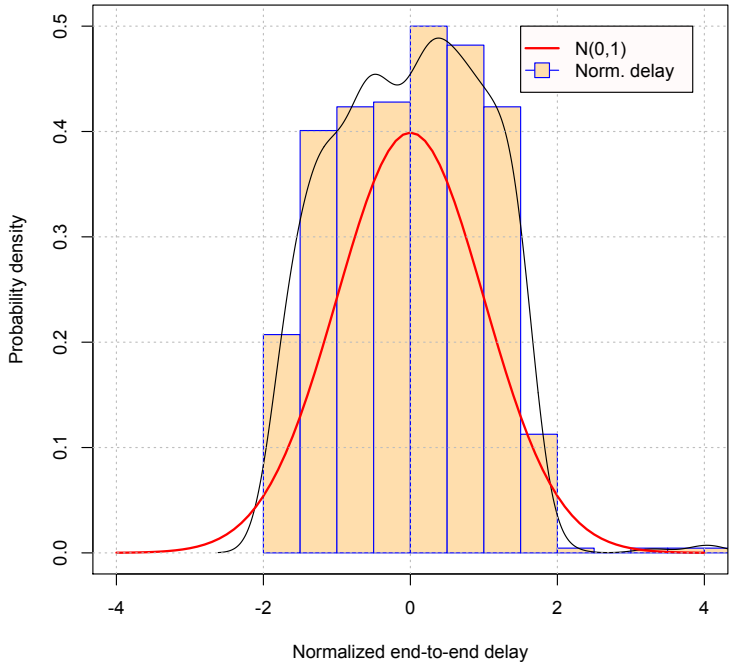


Figure VI.13: *Histogram and probability density function of end-to-end latency for a 1000-messages test-bed run in a 2-hop network as in scenario 1.*

The Probability Density Function (PDF) of the estimated end-to-end distribution in comparison with the reference distribution $N(0,1)$ shows an accurate match in all figures. The small error between the two lines accounts for a deviation which in most cases remains below 5%. This small inaccuracy is negligible for the application of the estimation method and particularly for the application of network trade-offs based on the relative evolution of the estimated distribution (i.e. analysis of trends).

VI.3 Evaluation Discussion

In TARP, the information carried out by control messages with respect to the estimated end-to-end distribution is made available to the application. Hence, the application updates its perception of the timeliness performance at regular intervals. Within one of such intervals, the application expectations are that the end-to-end delay distribution of the transmitted stream of messages follows the estimated distribution.

This information is not only exploitable by the application itself, but also a valuable run-time indicator of the changes on the timeliness performance. Hence, trade-off adaptations involving the timeliness metric can be based not only on the offline analysis of set-points, but also take into consideration the current network performance evaluated at run-time.

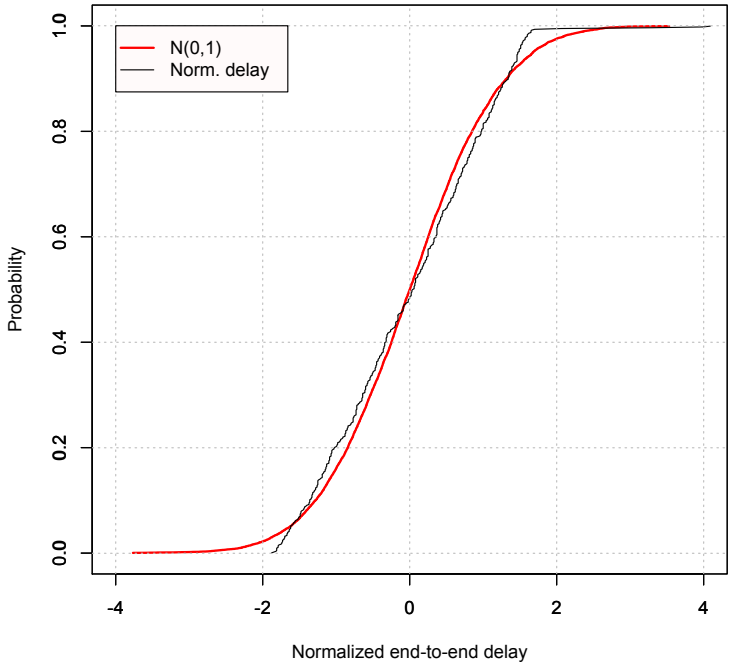


Figure VI.14: *Cumulative distribution function of end-to-end latency for a 1000-messages test-bed run in a 2-hop network as in scenario 1.*

The evaluation of scenarios presented in this chapter shows that in both environments the method produces satisfactory results. The experiments evaluated on a simple test-bed showed that the estimation algorithm is accurate on a real implementation, which in some cases exceeded the performance observed in simulations. Despite significant efforts were put to minimize the difference between abstract models and the real platform, this outperformance shows that in some cases the simulation models were slightly pessimistic.

The overhead introduced by this method is minimal as it takes advantage of the transmission of regular messages for the estimation of local forwarding delays. The timeliness aware protocol presented in chapter V is based on a tree-routing algorithm, although the adaptation of different routing strategies is possible with limited efforts.

Additional experiments were carried out during the evaluation of the work presented in this thesis. Some of the most representative are included in appendix B.

VI.4 Chapter summary

This chapter presented the evaluation of concepts introduced in the thesis. Simulations and deployed test-beds validate the concepts and provide a graphic evaluation of the accuracy of the end-to-end distribution method presented in chapter III. The method is evaluated under a number of scenarios configured according to a meaningful variety of parameters. The timeliness aware routing protocol introduced in chapter V proves

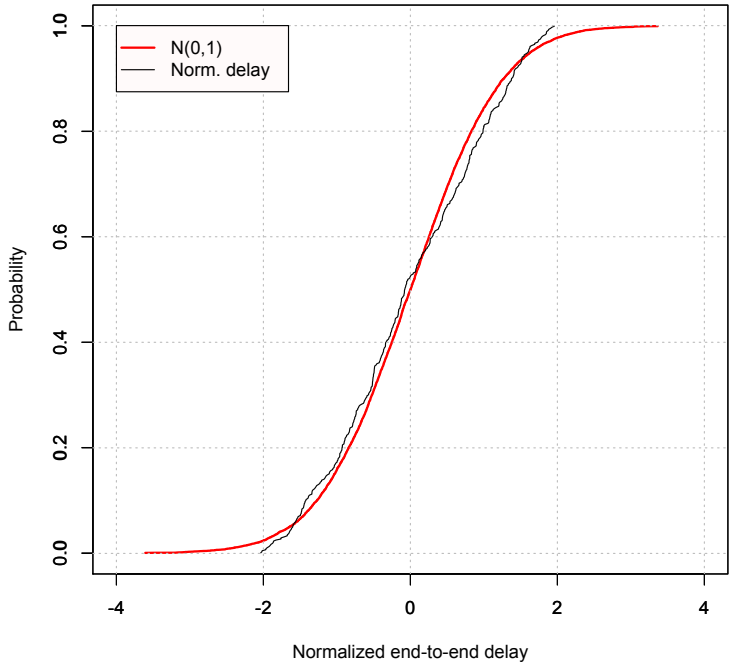


Figure VI.15: *Cumulative distribution function of end-to-end latency for a 1000-messages test-bed run in a 3-hop network as in scenario 1.*

to perform accurate estimations of the distribution of end-to-end latency at a negligible cost, hence enabling the application of network trade-offs with dynamic run-time information regarding the timeliness performance.

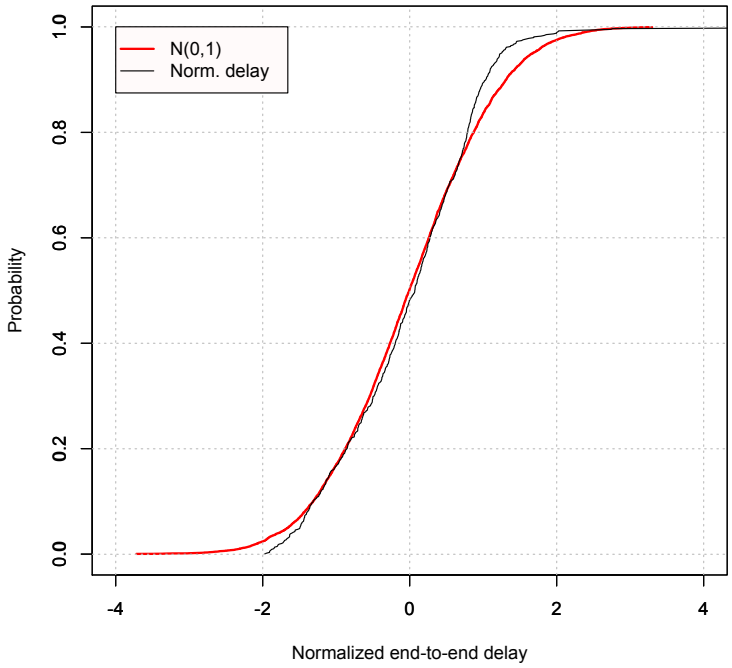


Figure VI.16: *Cumulative distribution function of end-to-end latency for a 1000-messages test-bed run in a 4-hop network as in scenario 1.*

Chapter VII

Conclusions

This chapter concludes the work presented along the thesis with an overview of the main contributions and final remarks.

VII.1 Overview of Main Contributions

The work presented in this thesis offers an alternative timeliness approach to WSNs. In doing so, it develops a *generalized timeliness notion* adapting to the particularities of WSNs. The new conceptual notion allows defining feasible real-time objectives without incurring in misleading system assumptions over-constraining the applicability of methods.

Exploring QoS *trade-offs* between timeliness and other significant QoS *metrics*, the core of this thesis is centered in the definition and application of *local* and *global trade-offs*. The association between timeliness and other related metrics enables the definition of alternative set-points influencing the quality performance of the network at selected instants of time.

A simple routing protocol embedding these concepts constitutes a proof for the validity of the presented concepts.

The main contribution of this thesis can be divided into three categories:

VII.1.1 Timeliness Notion

Chapter II overviewed the current state-of-the-art with particular interest in the state of real-time networking protocols. The appreciation of significant inconsistencies regarding the assumptions and objectives of existing real-time methods motivated the analysis of common misconceptions and misleading assumptions restricting the applicability of current real-time methods in realistic scenarios. Section II.5 contributes with a series of recommendations concluding the analysis of a number misconceptions identified within the analyzed literature.

Chapter III presented an alternative timeliness notion adapting to the particularities of WSNs. The generalized timelines notion is by itself a contribution as it opens a new range of possibilities to defining timeliness approaches. The main strength of this notion

falls upon the capacity of defining timeliness aware methods without incurring in the above-mentioned misleading assumptions.

As a proof of concept for the potential of the generalized notion, chapter III includes an algorithm estimating the end-to-end latency distribution of routing paths.

VII.1.2 Trade-off Analysis

The contribution in chapter IV starts with the analysis of local and global trade-offs. The chapter explores the definition of set-points based on the analysis of abstract models under a number of possible application modes. The analysis of set-points and trends defines the possibilities to exercise run-time network trade-offs, provided that the necessary hooks are available.

Chapter V describes a simple routing protocol based on the trade-off analysis as well as the generalized timeliness notion.

VII.1.3 Operating System

In addition to the contributions regarding network communications, this thesis also contributes in the definition of the OSAL. This level of abstraction allows a unified Operating System (OS) enabling a wider flexibility in the exploitation of local trade-offs and the definition of portable applications.

VII.2 Final Remarks

During the evaluation of the work presented in this thesis a large number of test-bed and simulation runs were carried out. The difficulties in finding a reasonable balance between the results observed from small test-bed deployments and simulated scenarios is well worth a mention in the final remarks. In particular, a significant amount of time and efforts were dedicated to match the characteristics of the two evaluation environments and make sense of the results obtained. In some cases this mismatch was a valuable clue in the process of debugging, but not only the simulation models were fixed according to the results obtained in real deployments. Anecdotically, thanks to the debugging possibilities included in the simulation environment, it was possible to detect a number of bugs in the test-bed implementation.

Appendix A

OSAL: An Operating System Abstraction Layer

The establishment of proper control mechanisms for the hardware platform into software frameworks brings up a number of portability issues. Operating System Abstraction Layer (OSAL) [Schoofs 09], [Serna Oliver 10c] has been designed to address these issues and diminish the conflicts between different software and hardware platforms. In particular, it addresses the discrepancies among different Operating System (OS) with respect to their functional Application Programming Interfaces (API), hardware configuration mechanisms, resource management and handling of peripherals.

A.1 Abstraction of the Operating System

OSAL is an abstraction layer designed to be placed on top of an OS, which translates system primitives from the original operating system into an unified API. Thus, application builders are able to use a common API which has the advantage of dramatically reducing portability efforts in later deployments. Experience shows that after the first implementation, porting OSAL to new platforms requires a relatively small effort compared to a large scale deployment [Serna Oliver 10c].

OSAL achieves minimal footprint and execution overhead by using advanced compilation tools and techniques. Among others, function-level linking, in-line functions and extensive use of preprocessor macros to provide light function wrappers for the OS native primitives.

The main goals of OSAL include the unification of basic data types and structures, a common API of the system primitives and return codes, extension or implementation of non-compliant or missing primitives, and abstraction of specific OS and hardware initialization procedures.

A.1.1 OSAL API

OSAL API covers the basic needs to develop applications in the domain of sensor networks. This chapter presents the representative parts of its design with appropriate examples for each case.

OS Initialization

Start-up initialization is one the most system-dependent features in any OS. For example, FreeRTOS [RTE 09] leaves this responsibility to the entry function, which normally creates some tasks before calling `vTaskStartScheduler()` to trigger the OS scheduler. This primitive does not return, and hence the initial task itself is permanently suspended.

In contrast, MANTIS OS initializes the scheduler before calling the application-defined entry function (`void start()`), where any additional required task can be created.

OSAL defines a initialization function (`wos_main()`) which always runs after the scheduler has been initialized. Any task created in this function will be scheduled immediately.

Table A.1 summarizes the initialization procedure for the three systems.

Tasks

The OSAL API unifies the way in which parameters are given to the OS and the particularities of each individual system.

Creating a task using OSAL is done with the primitive:

```
wos_task_create(thread_func, NULL)
```

The second parameter can be either `NULL` or a pointer to an initialized `task_attr_t` structure that allows specifying stack size and priority explicitly.

Synchronization

Primitives to ensure mutual exclusion between concurrent tasks are frequently used in embedded OS. OSAL provides support for mutex, semaphores and message queues, which cover most application requirements with respect to synchronization and inter-task communication.

Table A.11 shows the primitives for mutex handling of the two analyzed OSs for comparison with the OSAL API. Note that the FreeRTOS synchronization API has several critical drawbacks (e.g. impossibility to immediately wake up a thread of the same priority which was waiting for a semaphore). For this reason, we developed an additional *Advanced Synchronization Framework* with similar syntax which solved these issues.

Mantis OS

```
void start() {
   // Application initialization code
   // Application main thread
}
```

FreeRTOS

```
void MainThread() {
   // Application main thread
   vTaskDelete(xTaskGetCurrentTaskHandle());
}
int main() {
   vPortInitialize();
   // Application initialization code
   xTaskCreate(MainThread, NULL, main_stack_size,
               NULL, main_priority, NULL );
   vTaskStartScheduler();
   return 0;
 }
```

OSAL

```
wos_status wos_main(void) {
   // Application initialization code
   // Application main thread
   return MAKE_STATUS(WOS_SUCCESS);
}
```

Table A.1: *System initialization of an empty application in MANTIS OS, FreeRTOS and OSAL*

Software Timers

Software timers are of special importance in sensor network systems as they allow implementation of time-outs as well as periodic execution of functions, which are essential to any networking protocol. Efficient implementations use hardware timer interrupts to callback a given timer function. This, however, implies a number of limitations due to the execution within an interrupt context (ISR). Namely, that interrupts are disabled while a software timer handler is running and the *current task* context is undefined, invalidating the use of blocking functions.

The following example shows how software timers are handled in OSAL:

```
timer_t timer;
wos_timer_create(&timer, period, hdl_funct, context);
```

Mantis OS

`mos_mutex_t mtx;`
`mos_mutex_init(&mtx);`
`mos_mutex_lock(&mtx);`
`mos_mutex_unlock(&mtx);`

FreeRTOS

`xQueueHandle mtx = xQueueCreateMutex()`
`xSemaphoreTake(mtx);`
`xSemaphoreGive(mtx);`

OSAL

`wos_mutex_t mtx;`
`wos_mutex_init(&mtx);`
`wos_mutex_lock(&mtx);`
`wos_mutex_unlock(&mtx);`

Table A.2: *Synchronization primitives in MANTIS OS, FreeRTOS and OSAL*

```
wos_timer_destroy(&timer);
```

Note that FreeRTOS does not originally support software timers. The implementation of this functionality on top of FreeRTOS was necessary before porting the OSAL API.

A.1.2 Message Handling

Efficient handling of messages is vital to sensor applications. OSAL unifies the most relevant aspects regarding radio transmissions of messages, namely message structures, buffers and radio API.

Packet Buffers

As different OS use different structures for packet buffers (e.g. some of them are fixed-size, some support variable length), OSAL provides a unified preprocessor macro allowing to declare a structure which specifies its name and the desired size.

For example, OSAL provides procedures to declare packet buffers and structures both for sending and receiving packets (i.e. `DECLARE_PACKET(name, max_size)`). Particular constrains may appear due to native restrictions on the underlaying OS. For instance, the parameter `max_size` must be checked and a compilation error raised if its value exceeds the maximum supported size by the OS.

Radio API

The radio API is a subset of OSAL which abstracts internal structures and the management of data packets. Packet buffers can be both of fixed or variable size, and can contain (or not) several internal fields. The goal of the radio API is to abstract platform-specific details from the user and to provide a transparent interface for dealing with packets.

Particular attention has been given to the buffer management for radio transmission and reception. For example, MANTIS OS copies incoming packets into a fixed system-maintained buffer (i.e. a packet is always received in the same place and then can be copied by the application). OSAL provides a preprocessor macro, which enables direct access to the system buffers, avoiding costly and unnecessary memory operations.

MANTIS OS provides relatively inconvenient API for sending and receiving radio packets. A packet should be contained in a fixed-size buffer structure with a field `size`directly set by the primitive caller. A typical scenario involving sending some fixed-structure packets using MANTIS API is the following:

```
struct VerySimpleMessage {
   unsigned long OrderNumber;
};
comBuf buf;
buf.size = sizeof(VerySimpleMessage);
((VerySimpleMessage *)buf.data)->OrderNumber = n;
com_send_IFACE_RADIO(&buf);
```

OSAL provides a more convenient object-oriented API for sending radio packets, reducing the previous code to the following:

```
EnclosingPacket<VerySimpleMessage> buf;
buf->OrderNumber = n;
Radio::SendPacket(buf);
```

Moreover, OSAL hides all OS-specific low-level details, such as fixed-size buffers, which enables straight forward portability among different OS.

A.1.3 Build System

Developing almost every embedded application starts with setting up the build environment. One of the goals of the OSAL is to simplify build system-related tasks as much as possible.

The steps required to build an application differ for the two analyzed OSs. Under MANTIS OS, the user needs to create an `automake` file to generate a `Makefile`, which must be re-generated when new sources are added. No built-in debug/release configuration support is provided, although this can be done by editing the generated `Makefile`. Generally, a typical `automake` file consists of $3 - 4$ lines.
FreeRTOS does not provide a build system itself. Hence, users should manually create

`Makefile` and specify all involved sources and build flags. A typical FreeRTOS-based application `Makefile` consists of around 50 lines.

This work explored the possibilities of `GNU make` to provide a simple building system for OSAL. As a result, in includes a number of scripts which dramatically reduce the build system-related complexity, as shown in the following `makefile` example of building a simple application:

```
WOSMAKE_ROOT = ../../../../makesystem
include $(WOSMAKE_ROOT)/Makeapp.lazy
```

In this case, the binary file name and the source list is automatically generated based on current directory name and contents. Alternatively, it is possible to specify this information explicitly:

```
WOSMAKE_ROOT = ../../../../makesystem
wosapp_objects = my_obj1.o my_obj2
wosapp_image = my_binary
```

Note that the OSAL build system automatically supports switching between MANTIS and FreeRTOS, as well as between `DEBUG` and `RELEASE` configurations, requiring no makefile or source file modifications.

A.2 Evaluation

The OSAL has been successfully implemented on top of MANTIS OS and FreeRTOS as part of the EU funded project WASP [Consortium 10a] with the internal name *WASP OS API* (WOS). Complete documentation on the design and implementation process is publicly available and periodically updated in [Serna Oliver 10b] as well as in public deliverables under *Work Package 3* in [Consortium 10a].

A.2.1 Code and Memory Footprint Overhead

This section presents the evaluatuon of the overhead on the executable code size and the memory footprint of initialized and non-initialized memory.It identifies the overhead of each basic primitive and extrapolate the run-time overhead based on the composition of these measurements. The implementation ensures that no additional run-time overhead exists due to the abstraction layer.

The evaluation is based on a number of sample applications to easily identify when and why overhead was generated comparing four OS configurations: MANTIS OS, MANTIS OS + OSAL, FreeRTOS, and FreeRTOS + OSAL.

OS Initialization Overhead

Table A.4 shows the evaluation of the start-up initialization functions in Table A.1. In this example we compare the byte sizes for the compiled code in `RELEASE` and `TRACE` modes. In the latter, OSAL initializes the *tracing framework* (mutex and some buffers), responsible for the additional overhead compared to the former.

The difference between the two modes is due to the availability of debug tracing as, e.g. `wos_dbg_print("Bug!");`. This allows inserting debugging lines that only produce binary code in `TRACE` mode, resulting in no overhead when compiling in `RELEASE` mode.

Our evaluation shows a minimal overhead of 22 bytes in Mantis OS and 40 bytes in FreeRTOS due to the initialization process with much lower overhead on the memory footprint in `RELEASE` mode. Note that the impact of the `TRACE` configuration is relatively small, enabling the use of debugging primitives at a reasonable cost.

Starting from the synchronization example, we will use the OSAL initialization function for both raw RTOS and OSAL-based builds to filter out the constant initialization overhead and to track only the changes implied by using OSAL API instead of raw RTOS API.

Task Handling Overhead

The evaluation of the overhead added to the task handling primitives considers the code shown in table A.3. Table A.5 shows that the overhead is mostly the same as for previous sample which is caused by the initialization functions. To distinguish between startup-related overhead and other types of overhead, we change the FreeRTOS and MANTIS samples to use the OSAL initialization while directly calling the original OS primitives. The modified code is shown in Table A.10.

Note that when FreeRTOS is used, OSAL wraps all thread functions in an own function that calls `vTaskDelete()` after a thread function has returned. This produces a constant 6 byte overhead plus 2 bytes for each task being created.

Further examples in the rest of this section, do not include the initialization overhead in order to provide a better overview of the the isolated overhead of each evaluated functionality.

Synchronization overhead

The evaluation of the overhead of synchronization primitives is done by comparing the footprint sizes of a relatively small program that performs iterative mutex testing, as shown in Table A.11.

Table A.8 shows that OSAL does not introduce any overhead for MANTIS OS, and that the only overhead for FreeRTOS are the already mentioned 6 bytes plus 2 additional bytes for each task created, which are due to the task function wrapping. Therefore, the synchronization API itself does not produce any additional overhead, neither in code size nor in memory footprint.

Timers Overhead

The evaluation of the overhead introduced in software timers is done comparing the footprint sizes for a sample application registering three timer handlers which increase one of three static variables. Again, as OSAL functions are simple inline wrappers around the native OS calls, no overhead is produced as shown in Table A.7.

Radio API Overhead

With respect to the radio API, Table A.12 shows a simple application sending a sequence of short messages. Since FreeRTOS does not directly support radio interfacing this part only evaluates this feature with MANTIS OS. Table A.9 shows that the usage of OSAL primitives does not increment the footprint, hence incurring in no additional overhead due to OSAL.

A.2.2 Evaluation Overview

Table A.13 summarizes the additional overhead introduced by OSAL for each of the analyzed functionalities. Note that the overhead, measured in bytes, is accounted for each of the referred functions in isolation (e.g. ignoring the overhead due to initialization). The total additional overhead of a complex application depends on the number of tasks and other resources created, although our analysis shows that it is negligible with respect to the overall footprint of a real application.

Mantis OS

```
#include <mos.h>
#include <msched.h>
#include <led.h>
#define DEFAULT_STACK_SIZE 128

void thread1(void) {}
extern "C"
void start(void) {
   mos_thread_new( thread1, DEFAULT_STACK_SIZE,
                   PRIORITY_NORMAL);
}
```

FreeRTOS

```
#include "FreeRTOS_all.h"

void thread2(void *) {
   vTaskDelete(xTaskGetCurrentTaskHandle());
}
void thread1(void *) {
   FreeInitialThreadStack();
   xTaskCreate(thread2, NULL, 64, NULL, 1, NULL);
   vTaskDelete(xTaskGetCurrentTaskHandle());
}
int main() {
   FreeRTOS_InitializeForWASP();
   xTaskCreate(thread1, NULL, 64, NULL, 1, NULL);
   vTaskStartScheduler();
}
```

OSAL

```
#include <wos/task.h>
#include <wos/led.h>

void thread1(void) {}
wos_status wos_main(void) {
   wos_task_create(thread1, NULL);
   return MAKE_STATUS(WOS_SUCCESS);
}
```

Table A.3: *Simple application example creating one empty task*

	Size	Mem-I	Mem-NI
MANTIS	9048	20	534
MANTIS+OSAL(R)	9056 (+8)	20	534
MANTIS+OSAL(T)	9070 (+22)	20	544 (+10)
FreeRTOS	4448	46	114
FreeRTOS+OSAL(R)	4474 (+24)	46	114
FreeRTOS+OSAL(T)	4488 (+40)	46	116 (+2)

Table A.4: *Overhead introduced by OSAL in system initialization functions (see Table A.1) compiled in different modes:* `TRACE` *(T) and* `RELEASE` *(R).*

	Size	Mem-I	Mem-NI
MANTIS	9066	20	534
MANTIS+OSAL	9074 (+8)	20	534
FreeRTOS	4480	46	114
FreeRTOS+OSAL	4510 (+30)	46	114

Table A.5: *Overhead introduced by OSAL in task creation functions (see Table A.3).*

	Size	Mem-I	Mem-NI
MANTIS	9088	20	544
MANTIS+OSAL	9088	20	544
FreeRTOS	4518	46	116
FreeRTOS+OSAL	4526 (+8)	46	116

Table A.6: *Overhead introduced by OSAL in task creation functions with original OS primitives (see Table A.3).*

	Size	Mem-I	Mem-NI
MANTIS	10842	20	679
MANTIS+OSAL	10842	20	679
FreeRTOS	6974	46	210
FreeRTOS+OSAL	6974	46	210

Table A.7: *Overhead introduced by OSAL in a sample timer application.*

	Size	Mem-I	Mem-NI
MANTIS	11324	32	639
MANTIS+OSAL	11324	32	639
FreeRTOS	6974	58	164
FreeRTOS+OSAL	6986 (+12)	58	164

Table A.8: *Overhead introduced by OSAL in synchronization functions (see Table A.11).*

	Size	Mem-I	Mem-NI
MANTIS	11498	22	633
MANTIS+OSAL	11498	22	633

Table A.9: *Overhead introduced by OSAL in a sample message sending application (see Table A.12).*

Mantis OS

```
#include <wos/task.h>
#define DEFAULT_STACK_SIZE 128

void thread1(void) {}
wos_status wos_main(void) {
   mos_thread_new(thread1, DEFAULT_STACK_SIZE,
                  PRIORITY_NORMAL);
   return  MAKE_STATUS(WOS_SUCCESS);
}
```

FreeRTOS

```
#include <wos/task.h>

void thread2(void *) {
   vTaskDelete( xTaskGetCurrentTaskHandle());
}
wos_status wos_main(void) {
   xTaskCreate(thread2, NULL, 64, NULL, 1, NULL);
}
```

Table A.10: *Modified simple application example creating one empty task with original OS primitives*

```
#include <wos/task.h>
// ...
static wos_mutex_t s_TestMutex;
static int s_Busy = 0;
static int s_Error = 0;
static int s_Iter = 0;
static int s_Threads = 0;
#define MAX_SLEEP_T  100

void thread_body(void) {
   int i;
   wos_mutex_lock(&s_TestMutex);
   s_Threads++;
   wos_mutex_unlock(&s_TestMutex);
   for (i = 0;;i++) {
      wos_mutex_lock(&s_TestMutex);
      if (s_Busy) {
         WOS_LED_ON(RED);
         wos_dbg_print("Bug!");
         s_Error = 1;
      }
      s_Busy = 1;
      wos_sleep(((unsigned)rand()) % MAX_SLEEP_T);
      s_Iter++;
      s_Busy = 0;
      wos_mutex_unlock(&s_TestMutex);
   }
}
wos_status wos_main(void) {
   wos_mutex_init(&s_TestMutex);
   wos_task_create(thread_body, NULL);
   wos_task_create(thread_body, NULL);
   wos_task_create(thread_body, NULL);
   for (;;) {
      wos_dbg_printf("Iteration: %5d;
                     threads: %d; error: %d\n",
                     s_Iter, s_Threads, s_Error);
   }
   return MAKE_STATUS(WOS_SUCCESS);
}
```

Table A.11: *Sample OSAL application using synchronization primitives.*

```
struct VerySimpleMessage {
   unsigned long OrderNumber;
};
using namespace WOS::Radio;
wos_status wos_main(void) {
   wos_radio_init();
   EnclosingPacket<VerySimpleMessage> buf;
   for (unsigned long n = 1;;n++) {
      buf->OrderNumber = n;
      wos_dbg_printf("Sending... (%ld)", n);

      WOS_LED_ON(GREEN);
      Radio::SendPacket(buf);
      wos_dbg_printf("done\n");
      WOS_LED_OFF(GREEN);
      wos_sleep(100);
   }
   return MAKE_STATUS(WOS_SUCCESS);
}
```

Table A.12: *Sample OSAL application sending a sequence of short messages containing increasing numbers.*

	MANTIS OS			**FreeRTOS**		
Function	(a)	(b)	(c)	(a)	(b)	(c)
Initialization	+8	0	0	+24	0	0
Tasks	0	0	0	+8	0	0
Mutex	0	0	0	+12	0	0
Timers	0	0	0	0	0	0
Messages	0	0	0	-	-	-

Table A.13: *Summary of overhead introduced by OSAL (in bytes) in (a) code size, (b) initialized memory, and (c) non initialized memory.*

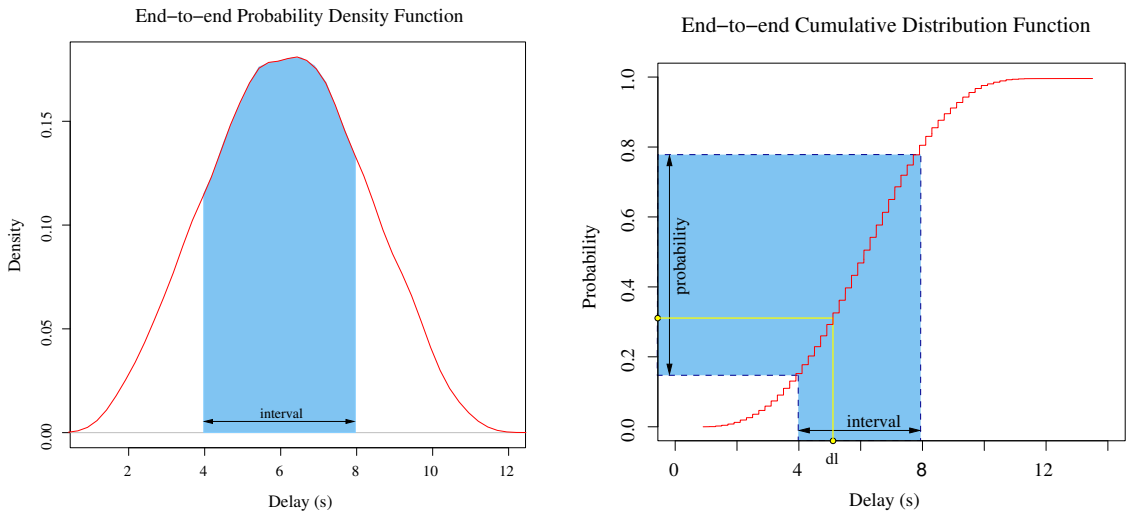
Appendix B

Extended results

B.1 Additional Example

This example is based on a typical Wireless Sensor Network (WSN) with a routing path of length q-hops as depicted in figure III.6. Each hop in the path is logically connected to two additional neighbors through in the routing tree. Note that the physical interference range is not limited by the logical connection and –as it happens in real deployments– interferences between nearby transmitting nodes may occur.

Following this set-up, figure B.1 shows the probability density function (PDF) obtained by simulation of a routing path of length 5. Each node on the network (including those forming the path) generated traffic with a time between messages following an exponential distribution with parameter $\lambda = 15s$.



(a) *PDF of the of the end-to-end latency distribution showing the target interval*

(b) *CDF of the of the end-to-end latency distribution showing target interval and probability*

Figure B.1: *Expressing timeliness by means of the end-to-end latency distribution (PDF and CDF).*

The artificial load was set to simulate the effects of cross-traffic on a segment of a

big network. Additional control messages were generated periodically every $30s$ at the source of the path and its end-to-end delay captured upon their arrival at the sink.

This scenario was simulated by means of the network simulator Omnet++ [Varga 01] [Community 10] and Mobility Framework [Koepke 10]. The chosen MAC protocol was Wisemac [El-Hoiydi 04] and the radio model followed the specifications of a CC2420 transceiver [cc2 08]. The routing path was manually fixed for this experiment and all messages in the network were directed to the sink.

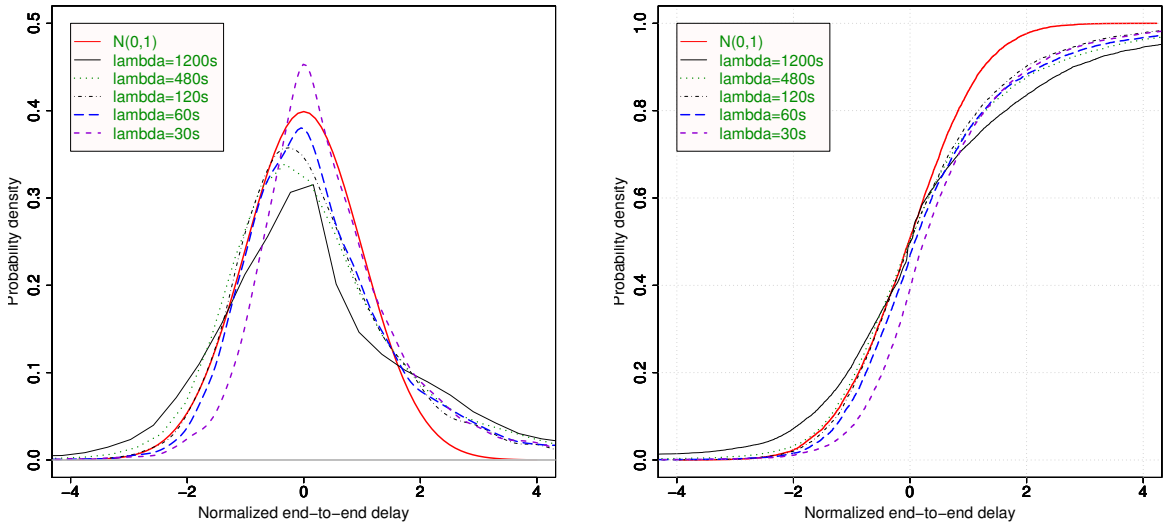
In this example, the timeliness requirements correspond to the interval $(4s, 8s)$. Hence, the area bellow the pdf curve represents the probability of end-to-end delays to fall within the interval. At run-time, it is possible to analyze the percentage of messages from a sequence fulfilling this timeliness requirement.

Figure B.1b depicts the estimated cumulative distribution function of the same experiment with an additional line illustrating the classic strict timeliness notion (dl). The probability of fulfilling the timeliness requirements is highlighted and represents approximately 60%.

Both figures, illustrate the relation between the bounds of the time interval and the achieved probability. Note that both factors are directly dependent of each other.

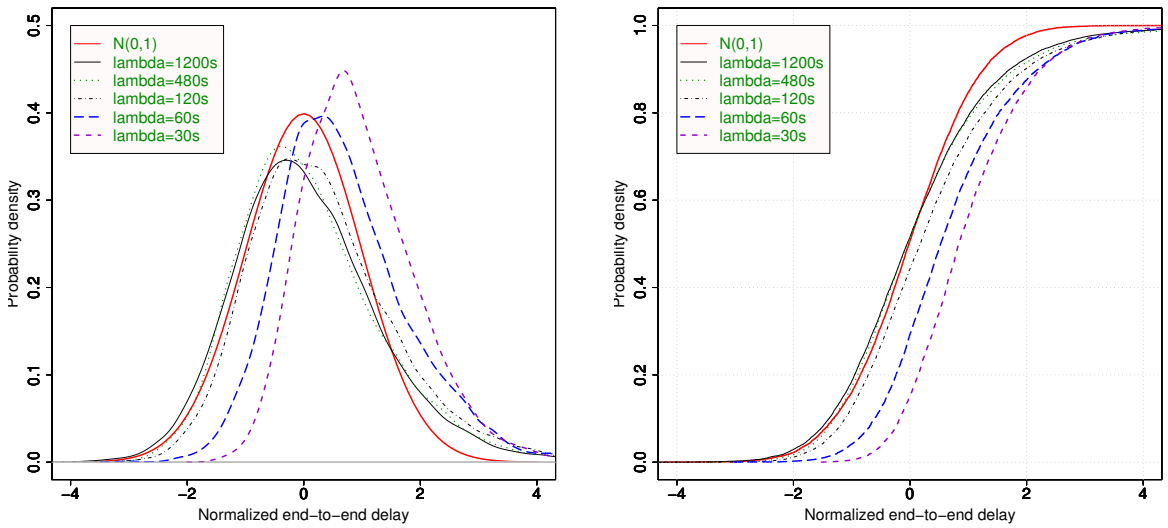
B.2 Compact Representation of End-To-End Distribution in Scenario 2

Figures B.2 and B.3 present a compact view of the evaluation of scenario 2 analyzed in detail in chapter VI. For a description of the scenario and detailed analysis refer to the chapter.



(a) *End-to-end distributions (PDF)* **(b)** *End-to-end distribution (CDF)*

Figure B.2: *Estimated PDF and CDF corresponding to scenario 2 with* $|rp| = 5hops$.



(a) *End-to-end distributions (PDF)* **(b)** *End-to-end distribution (CDF)*

Figure B.3: *Estimated PDF and CDF corresponding to scenario 2 with* $|rp| = 10hops$.

Table B.1 extends the results for the second test introduced in chapter III for the case of $|rp| = 10$ with the reference to the standard Normal in brackets. Refer to the chapter for a detailed description of the test.

λ N(0,1)	I_1 (68%)	I_2 (27%)	I_3 (4.2%)	I_4 (0.2%)
30	55.7% (-12.3)	30.1% (+3.1)	11% (+6.8)	3.2% (+3)
60	62.4% (-5.6)	24.9% (+2.1)	9.2% (+5)	3.5% (+3.3)
120	62% (-6)	27.1% (+0.1)	7.4% (+3.2)	3.6% (+3.4)
480	61.4% (-6.6)	28.5% (+1.5)	6.9% (+2.7)	3.2% (+3)
1200	60.7% (-7.3)	28.9% (+1.9)	7.6% (+3.4)	2.8% (+2.6)

Table B.1: *Percentage of hits per σ-interval with path length* 10. *In brackets, deviation with respect to $N(0,1)$.*

B.3 Extended Test-Bed Evaluation of Scenario 1

Figures B.4, B.5, B.6, B.7, B.8, and B.9 extend the test-bed evaluation of scenario 1 presented in chapter VI. Refer to the chapter for a detailed description of the scenario and experiments.

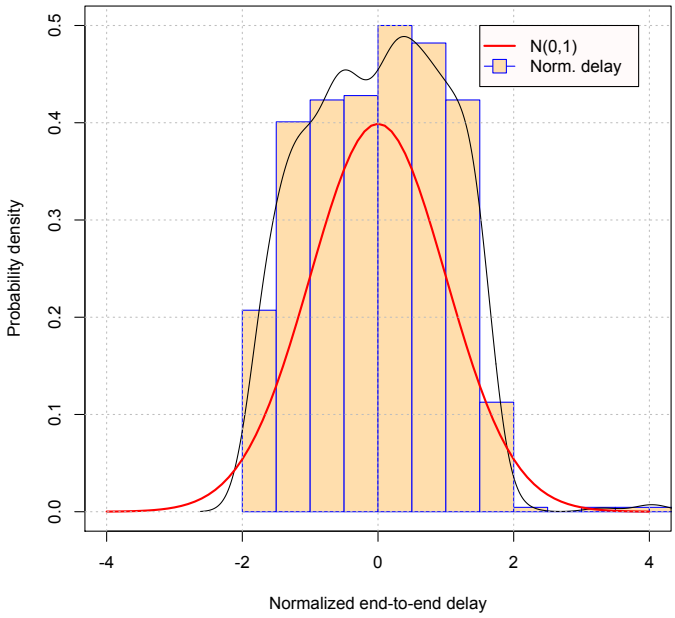


Figure B.4: *Histogram and probability density function of end-to-end latency for a 1000-messages test-bed run in a 2-hop network as in scenario 1.*

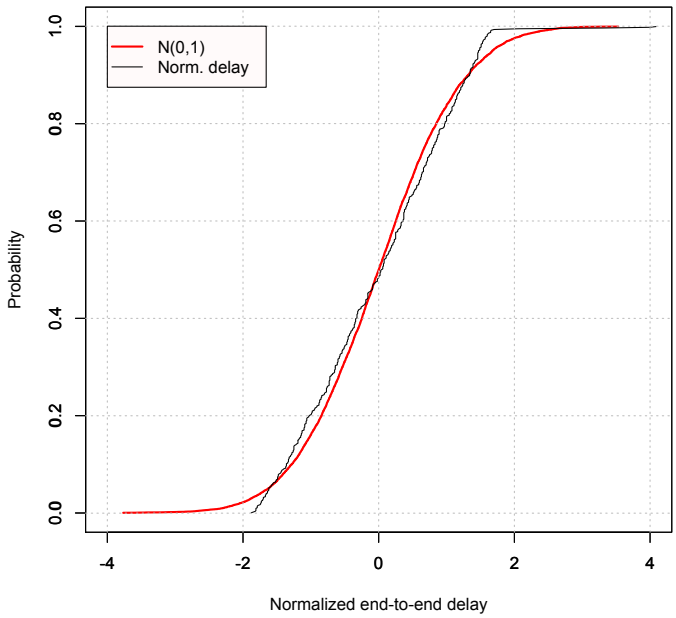


Figure B.5: *Cumulative distribution function of end-to-end latency for a 1000-messages test-bed run in a 2-hop network as in scenario 1.*

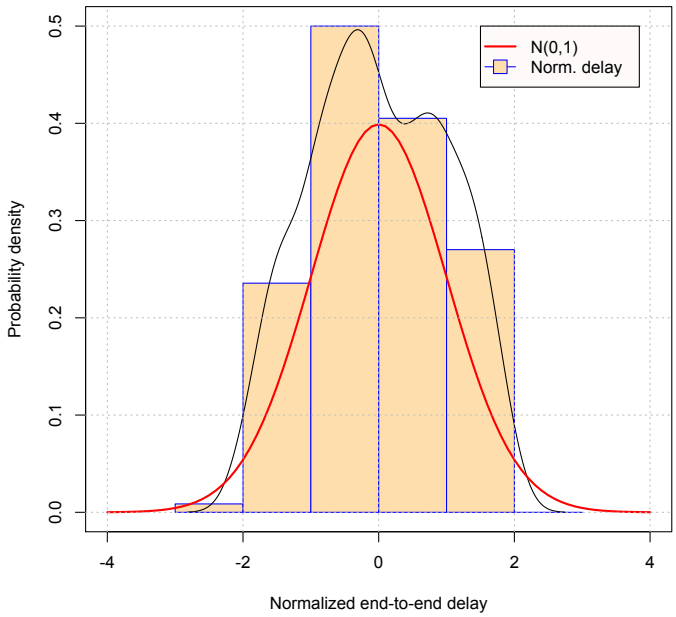


Figure B.6: *Histogram and probability density function of end-to-end latency for a 1000-messages test-bed run in a 3-hop network as in scenario 1.*

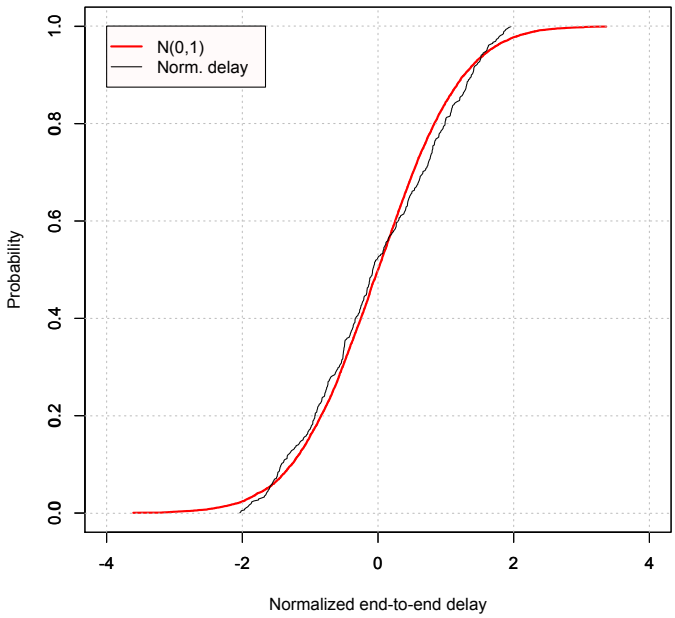


Figure B.7: *Cumulative distribution function of end-to-end latency for a 1000-messages test-bed run in a 3-hop network as in scenario 1.*

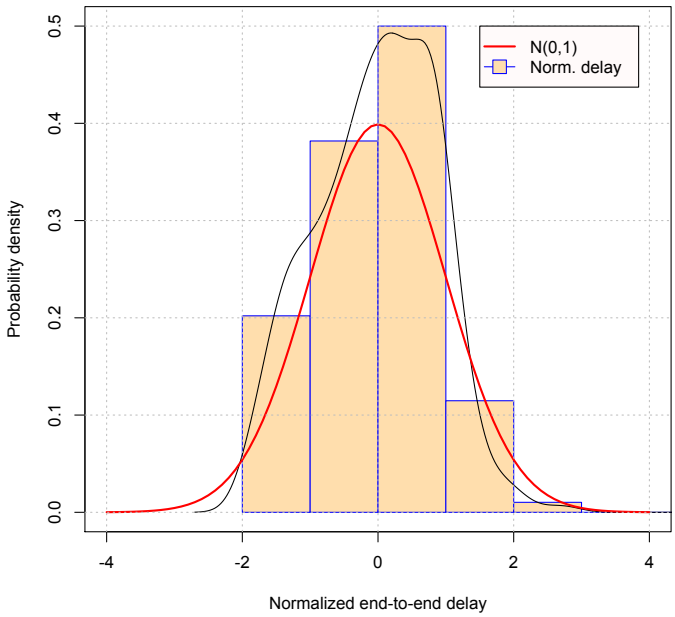


Figure B.8: *Histogram and probability density function of end-to-end latency for a 1000-messages test-bed run in a 4-hop network as in scenario 1.*

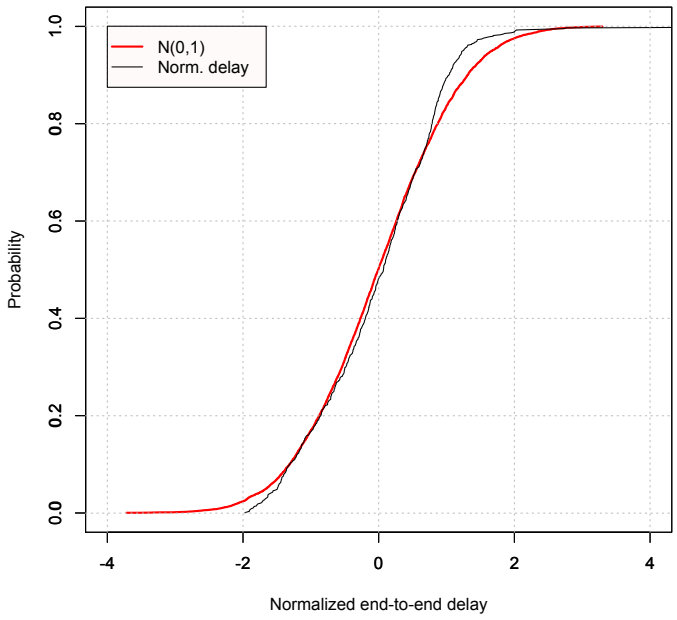


Figure B.9: *Cumulative distribution function of end-to-end latency for a 1000-messages testbed run in a 4-hop network as in scenario 1.*

B.4 Additional Evaluation Scenario: Arbitrary Network Deployment

This additional scenario is motivated from an elderly care use-case extracted from the WASP project [Lokhorst 07]. The goal of the depicted application is to provide automatized monitoring and medical assistance for elderly people living in a care house. Sensor nodes are attached to the patients and monitor their general health and well being, with special interest in aspects related to mobility and temperature.

The description of the scenario is as follows:

- approximately 20 to 30 patients living in the same home,
- nodes equipped with two sensors each: a 3D accelerometer and a temperature sensor,
- local processing of data is possible. The amount of transmitted data depends on the operational mode. In *normal mode*, nodes process their data locally and transmit at a low frequency (e.g. few messages per minute), whereas in the *special monitoring mode* the processing is much limited and messages are sent at a higher frequency (e.g. one message per second).

The configuration of parameters is derived from the elderly care scenario as follows:

- dimensions of a living room of approximately 20×10 meters,
- 30 nodes operating in *normal* mode, one in the special *monitoring* mode,

- sink is placed in one of the corners of the room,
- message payload is 64 bytes,
- traffic parameters as follows:
  - in normal mode: messages transmitted with a time between messages following an exponential distribution with parameter $\lambda = 30s$,
  - in special monitoring mode: messages transmitted periodically with period $T = 1s$,
  - control messages periodically broadcast by the sink with period $T = 60s$,

The following use-case provides a better understanding of the scenario.

Use-case example

Assuming a patient recovering from a major fall, the information from the accelerometer is constantly transmitted to monitor the patient's mobility. The locomotion analysis processes the accelerometer data, which acquires samples at a frequency ranging between 20 and 50Hz. Data aggregation is done locally to reduce the amount of traffic up to 1 message per second. Other nodes, of patients which do not require such special monitoring, process the sampled data by means of a local algorithm which generates an output at a much lower frequency (e.g. few messages per minute).

B.4.1 Simulation Results

Parameter	Simulation 1	Simulation 2
WiseMAC Sample Period	$50ms$	$50ms$
Topology	Scenario 3	Scenario 3

Table B.2: *Summary of the set-up for the simulation experiments*

Figures B.10 and B.11 show the normalized distributions (histogram and cdf) of a series of 5 simulations of the described scenarios, each of them limited to 100000 messages and one full day of simulated time. The standard distribution $N(0, 1)$ is plotted for comparison on each figure, which are cropped on the interval $(-4, 4)$ to highlight the relevant area. Nevertheless, only a few values exceeded the cropping point on the right side, following the expected behavior described in chapter III.

The application running on the sender node receives continuous information regarding the end-to-end timeliness performance. Hence, it can determine whether the probability of the following stream of messages to arrive within an acceptable time interval is satisfactory or not. In the latter, the application is made aware of the situation and can

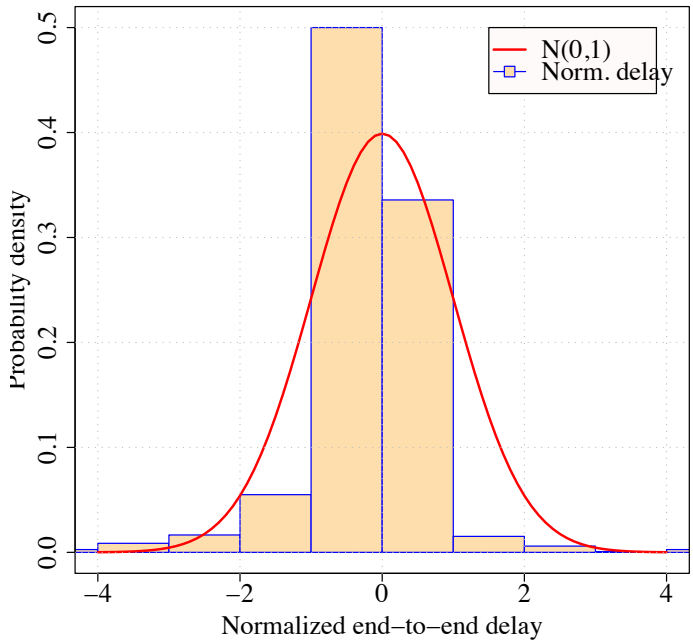


Figure B.10: *Evaluation of the Normalized end-to-end distribution histogram vs. PDF of* $N(0,1)$

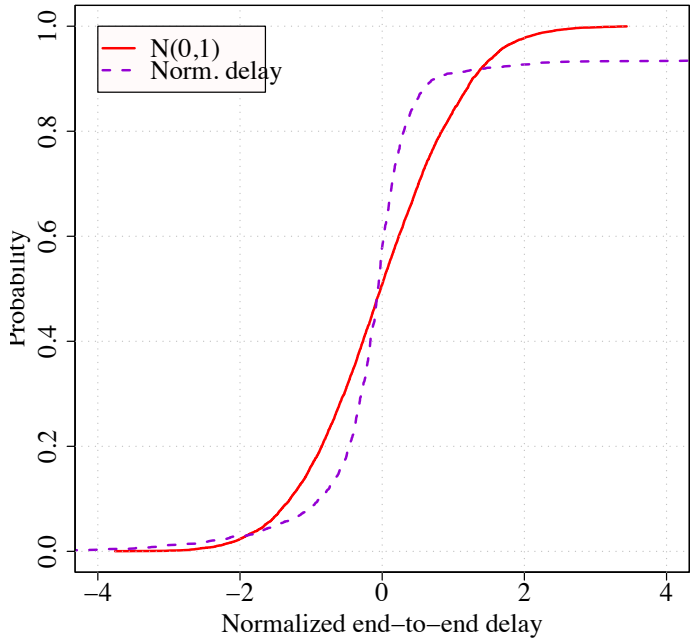


Figure B.11: *Evaluation of the Normalized end-to-end distribution CDF vs. CDF* $N(0,1)$

take adaptive measures (i.e. trade-off energy to perform more complex computations and reduce the amount of traffic).

Note that given the scenario, the length of routing paths remain most of the time between 2 and 3 hops. Comparing Figures III.4 and B.10, it is noticeable that the length of the path has an impact on the accuracy of the method. However, the results obtained for paths as short as two or three hops underline the usefulness of the method.

Bibliography

[Abdelzaher 03] T. Abdelzaher, J. Stankovic, S. Son, B. Blum, T. He, A. Wood & Chenyang Lu. *A communication architecture and programming abstractions for real-time embedded sensor networks*. In Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on, pages 220–225, May 2003.

[Abdelzaher 04] Tarek Abdelzaher, Shashi Prabh & Raghu Kiran. *On Real-time Capacity Limits of Multihop Wireless Sensor Networks*. In Proceedings of the IEEE International Real-Time Symposium (RTSS), 2004.

[Al-Karaki 04] J.N. Al-Karaki & A.E. Kamal. *Routing techniques in wireless sensor networks: a survey*. Wireless Communications, IEEE, vol. 11, no. 6, pages 6–28, Dec. 2004.

[Andersson 10] Björn Andersson, Paulo Gandra de Sousa, Filipe Pacheco, Panayiotis Andreou, Pedro Jose Marron, Umer Iqbal & Vinny Reynolds. *Elements of Scalable Data Processing*. In 9th International Workshop on Real-Time Networks RTN'2010, 2010.

[Benslimane 00] A. Benslimane. *Real-time multimedia services over Internet*. Universal Multiservice Networks, 2000. ECUMN 2000. 1st European Conference on, pages 253 –261, 2000.

[Berkeley 09] U.C. Berkeley. *TinyOS*. online, `http://www.tinyos.net/`, 2009.

[Bhatti 05] Shah Bhatti, James Carlson, Hui Dai, Jing Deng, Jeff Rose, Anmol Sheth, Brian Shucker, Charles Gruenwald, Adam Torgerson & Richard Han. *MANTIS OS: an embedded multithreaded operating system for wireless micro sensor platforms*. Mobile Networks and Applications (MONET) Journal, vol. 10, no. 4, pages 563–579, 2005.

[Birkel 93] Thomas Birkel. *A functional Central Limit Theorem for Positively Dependent Random Variables.* Journal of Multivariate Analysis, pages 314–320, 1993.

[Bulmer 67] M. G. Bulmer. Principles of statistics. Dover Publications, INC., 1979 edition, 1967.

[Burgstahler 02] Lars Burgstahler & Martin Neubauer. *New Modification of the Exponential Moving Average Algorithm for Bandwidth Estimation.* ITC Specialist Seminar, 2002.

[Burns 90] Alan Burns & A. J. Wellings. Real-time systems and their programming languages. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[Buttazzo 05] Giorgio C. Buttazzo. Hard real-time systems: Predictable scheduling algorithms and applications. Springer, second edition, 2005.

[Caccamo 02] M. Caccamo, L.Y. Zhang, Lui Sha & G. Buttazzo. *An implicit prioritized access protocol for wireless sensor networks.* In Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS), 2002.

[cc2 08] Cc2420: 2.4 ghz ieee 802.15.4 / zigbee-ready rf transceiver. Chipcon / Texas Instruiments, 2008.

[Cham 10] Jorge Cham. *Piled Higher and Deeper.* online, `http://www.phdcomics.com`, 2010.

[Chipara 06] O. Chipara, Zhimin He, Guoliang Xing, Qin Chen, Xiaorui Wang, Chenyang Lu, J. Stankovic & Tarek Abdelzaher. *Real-time Power-Aware Routing in Sensor Networks.* In Proceedings of the 14th IEEE International Workshop on Quality of Service (IWQoS)., June 2006.

[Community 10] OMNeT++ Community. *OMNeT++ Discrete Event Simulation Environment.* online, `http://www.omnetpp.org`, 2010.

[Consortium 10a] WASP Consortium. *EU Framework 6 IST Project "Wirelessly Accessible Sensor Populations" (WASP).* online, `http://www.wasp-project.org/`, 2006-2010.

[Consortium 10b] WASP Consortium. *WASP Tool-Chain.* online, `http://rts-wiki.eit.uni-kl.de/tiki-index.php?page=WOS+Framework`, 2010.

[Corporation 09] Sentilla Corporation. *Sentilla.* online, `http://www.sentilla.com`, 2009.

[Costa 07] N. Costa, A. Pereira & C. Serodio. *Virtual Machines Applied to WSN's: The state-of-the-art and classification.* In Second International Conference on Systems and Networks Communications, 2007. ICSNC 2007., pages 50 –50, 25-31 2007.

[Davis 07] Robert I. Davis, Alan Burns, Reinder J. Bril & Johan J. Lukkien. *Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised.* Real-Time Systems, vol. 35, no. 3, pages 239–272, 2007.

[De Couto 03] Douglas S. J. De Couto, Daniel Aguayo, Benjamin A. Chambers & Robert Morris. *Performance of multihop wireless networks: shortest path is not enough.* SIGCOMM Comput. Commun. Rev., vol. 33, no. 1, pages 83–88, 2003.

[de Freitas 08] E.P. de Freitas, M.A. Wehrmeister, C.E. Pereira & T. Larsson. *Real-time support in adaptable middleware for heterogeneous sensor networks.* In Computer Science and Information Technology, 2008. IMCSIT 2008. International Multiconference on, pages 593–600, Oct. 2008.

[Demirkol 06] I. Demirkol, C. Ersoy & F. Alagoz. *MAC protocols for wireless sensor networks: a survey.* IEEE Communications Magazine, vol. 44, no. 4, pages 115 – 121, April 2006.

[Duffy 07] Cormac Duffy, Utz Roedig, John Herbert & Cormac Sreenan. *An Experimental Comparison of Event Driven and Multi-Threaded Sensor Node Operating Systems.* In Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on, pages 267–271, March 2007.

[Dunkels 04] Adam Dunkels, Björn Gronvall & Thiemo Voigt. *Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors.* In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN04), pages 455–462, Washington, DC, USA, 2004. IEEE Computer Society.

[El-Hoiydi 04] Amre El-Hoiydi & Jean-Dominique Decotignie. *WiseMAC: An Ultra Low Power MAC Protocol for the Downlink of Infrastructure Wireless Sensor Networks.* In Proceedings of the Ninth IEEE Symposium on Computers and Communication (ISCC), June 2004.

[Ergen 06] Sinem Coleri Ergen & Pravin Varaiya. *PEDAMACS: Power Efficient and Delay Aware Medium Access Protocol for Sensor Networks.* IEEE Transactions on Mobile Computing, vol. 5, no. 7, pages 920–930, 2006.

[Ergen 07] Sinem Coleri Ergen & Pravin Varaiya. *Energy efficient routing with delay guarantee for sensor networks.* Wireless Networking, vol. 13, no. 5, pages 679–690, 2007.

[Felemban 05] E. Felemban, C.-G. Lee, E. Ekici, R. Boder & S. Vural. *Probabilistic QoS guarantee in reliability and timeliness domains in wireless sensor networks.* In Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM05, volume 4, pages 2646–2657, March 2005.

[Felemban 06] E. Felemban, Chang-Gun Lee & E. Ekici. *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, pages 738–754, June 2006.

[Gobriel 08] Sameh Gobriel, Robert Cleric & Daniel Mosse. *Adaptations of TDMA Scheduling for Wireless Sensor Networks.* In Proceedings of the 7th International Workshop on Real-Time Networks (RTN), 2008.

[Goldoni 08] Emanuele Goldoni & Giuseppe Rossi. *Improving Available Bandwidth Estimation using Averaging Filtering Techniques.* Rapport technique, Università degli Studi di Pavia, Laboratorio Reti, 2008.

[Guerra 09] Raphael Guerra & Gerhard Fohler. *A Gravitational Task Model with Arbitrary Anchor Points for Target Sensitive Real-Time Applications.* Real-Time Systems Journal, September 2009.

[Hadim 06] S. Hadim & N. Mohamed. *Middleware for Wireless Sensor Networks: A Survey.* In First International Conference on Communication System Software and Middleware, 2006. Comsware 2006., pages 1 –7, 0-0 2006.

[Hamdaoui 95] M. Hamdaoui & P. Ramanathan. *A dynamic priority assignment technique for streams with (m, k)-firm deadlines.* IEEE Transactions on Computers, vol. 44, no. 12, pages 1443 –1451, dec. 1995.

[He 03] T. He, J.A. Stankovic, C. Lu & Tarek Abdelzaher. *SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks.* In Proceedings of ICDCS'03, May 2003.

[He 05] Tian He, John A. Stankovic, Chenyang Lu & Tarek F. Abdelzaher. *A Spatiotemporal Communication Protocol for Wireless Sensor Networks.* IEEE Trans. Parallel Distrib. Syst., vol. 16, no. 10, pages 995–1006, 2005.

[Henricksen 06] Karen Henricksen & Ricky Robinson. *A survey of middleware for sensor networks: state-of-the-art and future directions.* In Proceedings of the international workshop on Middleware for sensor networks (MidSens '06), pages 60–65, New York, NY, USA, 2006. ACM.

[Herms 06] André Herms, Georg Lukas & Svilen Ivanov. *Realism in design and evaluation of wireless routing protocols.* In Proceedings of First international Workshop on Mobile Services and Personalized Environments (MSPE'06), 2006.

[Hill 02] Jason L. Hill & David E. Culler. *Mica: A Wireless Platform for Deeply Embedded Networks.* IEEE Micro, vol. 22, pages 12–24, 2002.

[Holland 06] M.M. Holland, R.G. Aures & W.B. Heinzelman. *Experimental investigation of radio performance in wireless sensor networks.* In Wireless Mesh Networks, 2006. WiMesh 2006. 2nd IEEE Workshop on, pages 140 –150, Sept. 2006.

[ICL 10] ICL. *BSN Node v.3.* online, `http://vip.doc.ic.ac.uk/bsn/a1875.html`, 2010.

[ISO 10] International Organization for Standardization ISO. *ISO/OSI Model.* online, `http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip`, June 2010.

[Karl 06] Holger Karl & Andreas Willig. Protocols and architectures for wireless sensor networks. Wiley, 2006.

[Koepke 10] Andreas Koepke. *Mobility Framework: A Framework to support simulations of wireless and mobile networks within OMNeT++.* online, `http://mobility-fw.sourceforge.net/`, 2010.

[Kopetz 94] H. Kopetz & G. Grunsteidl. *TTP-a protocol for fault-tolerant real-time systems.* Computer, vol. 27, no. 1, pages 14 –23, jan 1994.

[Korkmaz 03] Turgay Korkmaz & Marwan Krunz. *Bandwidth-delay constrained path selection under inaccurate state information.* In Proceedings of the IEEE/ACM Transactions on Networking, June 2003.

[Kulkarni 04] S.S. Kulkarni & Mahesh Arumugam. *TDMA service for sensor networks.* In Distributed Computing Systems Workshops, 2004. Proceedings. 24th International Conference on, pages 604 – 609, March 2004.

[Kumar 09] P. Kumar, M. Gunes, Q. Mushtaq & B. Blywis. *A real-time and energy-efficient MAC protocol for wireless sensor networks.* In IFIP International Conference on Wireless and Optical Communications Networks, 2009. WOCN '09, pages 1 –5, April 2009.

[Levis 02] Philip Levis & David Culler. *Mate: A tiny virtual machine for sensor systems.* In ACM, editeur, Proceedings of the 10th international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2002), pages 85 – 95, New York, NY, USA, 2002.

[Li 04] Peng Li, Binoy Ravindran & E. Douglas Jensen. *Adaptive time-critical resource management using time/utility functions: Past, present, and future.* In IEEE Computer Society, editeur, Proceedings of the 28th Annual International Computer Software and Applications Conference - Workshops and Fast Abstracts - (COMPSAC'04), pages 12–13, Washington, DC, USA, 2004.

[Liu 73] C. L. Liu & James W. Layland. *Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment.* Journal of the ACM (JACM), vol. 20, no. 1, pages 46–61, 1973.

[Liu 00] Jane W. S. Liu. Real-time systems. Prentice Hall, 2000.

[Lokhorst 07] C. Lokhorst, R.M. de Mol, H. Wells, M. Mazzu, F. Frumento, A. Corongiu, W. Savio, A.H. Ipema, A. Sorniotti & L. Gomez. *Deliverable D6.2: Specified application requirements and prototype selection.* Rapport technique, September 2007.

[Lu 02] C. Lu, B. Blum, Tarek Abdelzaher, J. Stankovic & T. He. *RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks.* In Proceedings of the IEEE RTAS 2002, 2002.

[Mainwaring 02] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk & John Anderson. *Wireless sensor networks for habitat monitoring.* In Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA02), pages 88–97, New York, NY, USA, 2002. ACM.

[Maróti 04] Miklós Maróti, Branislav Kusy, Gyula Simon & Ákos Lédeczi. *The flooding time synchronization protocol.* In SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems, pages 39–49, New York, NY, USA, 2004. ACM.

[Mavromoustakis 09] Constandinos X. Mavromoustakis & Helen D. Karatza. *Real-time performance evaluation of asynchronous time division traffic-aware and delay-tolerant scheme in ad hoc sensor networks.* Inter-

national Journal of Communication Systems, vol. 23, no. 2, pages 167 – 186, Sep 2009.

[Mizanian 09] K. Mizanian, R. Hajisheykhi, M. Baharloo & A.H. Jahangir. *RACE: A Real-Time Scheduling Policy and Communication Architecture for Large-Scale Wireless Sensor Networks.* In Communication Networks and Services Research Conference, 2009. CNSR '09. Seventh Annual, pages 458–460, May 2009.

[msp 04] Msp430x12x mixed signal microcontroller. Texas Instruments Incorporated, 2001-2004.

[Newport 07] Calvin Newport, David Kotz, Yougu Yuan, Robert S. Gray, Jason Liu & Chip Elliott. *Experimental Evaluation of Wireless Simulation Assumptions.* Simulation, vol. 83, no. 9, pages 643–661, 2007.

[Pothuri 06] P.K. Pothuri, V. Sarangan & J.P. Thomas. *Delay-Constrained, Energy-Efficient Routing in Wireless Sensor Networks Through Topology Control.* In Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control, 2006. ICNSC '06., pages 35 –41, 0-0 2006.

[Rhee 08] Injong Rhee, A. Warrier, M. Aia, Jeongki Min & M.L. Sichitiu. *Z-MAC: A Hybrid MAC for Wireless Sensor Networks.* IEEE/ACM Transactions on Networking, vol. 16, no. 3, pages 511 –524, june 2008.

[Rivas 03] Mario Aldea Rivas & Michael González Harbour. *Evaluation of New POSIX Real-Time Operating Systems Services for Small Embedded Platforms.* In Proceedings of the 15th Euromicro Conference on Real-Time Systems (ECRTS), 2003.

[Rossi 10] Dario Rossi. *Sensors as Hardware Motes Evolution.* online, `http://www.telematica.polito.it/wsn/ppt/WSN1_Hardware.pdf`, 2010.

[Rousselot 09] Jerome Rousselot, Jean-Dominique Decotignie, Marc Aoun, van der Stok Peter, Ramon Serna Oliver & Gerhard Fohler. *Accurate Timeliness Simulations for Real-Time Wireless Sensor Networks.* In European Modelling Symposium 2009, Athens, Greece, November 2009.

[RTE 09] Real Time Engineers Ltd RTE. *FreeRTOS.* online, `http://www.freertos.org/`, 2009.

[Sahoo 07] A. Sahoo & P. Baronia. *An Energy Efficient MAC in Wireless Sensor Networks to Provide Delay Guarantee.* In Proceedings of

the 15th IEEE Workshop on Local and Metropolitan Area Networks. LANMAN, June 2007.

[Schoofs 09] Anthony Schoofs, Marc Aoun, Peter van der Stok, Julien Catalano, Ramon Serna Oliver & Gerhard Fohler. *On Enabling Portable and Time-Controlled Wireless Sensor Network Applications*. In Proceedings of the 1st International Conference on Sensor Networks Applications, Experimentation and Logistics (SENSAPEAL09), Athens, Greece, September 2009.

[Serna Oliver 08] Ramon Serna Oliver & Gerhard Fohler. *Probabilistic Routing for Wireless Sensor Networks*. In Proceedings of Work-in-Progress Session, 29th IEEE Real-Time Systems Symposium 2008, Barcelona, Spain, December 2008.

[Serna Oliver 09a] Ramon Serna Oliver. *Estimation of the probability density function of end-to-end delays in Wireless Sensor Networks*. Rapport technique, Technische Universität Kaiserslautern, 2009.

[Serna Oliver 09b] Ramon Serna Oliver & Gerhard Fohler. *Probabilistic Estimation of End-to-End Path Latency in Wireless Sensor Networks*. In Proceedings of the Sixth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS09), Macau SAR, P.R.C, October 2009.

[Serna Oliver 09c] Ramon Serna Oliver & Gerhard Fohler. *A Proposal for a Notion of Timeliness in Wireless Sensor Networks*. In Proceedings of the 8th International Workshop on Real-Time Networks RTN'09, Dublin, Ireland, June 2009.

[Serna Oliver 09d] Ramon Serna Oliver, Ivan Shcherbakov & Gerhard Fohler. *Poster abstract: An Efficient Operating System Abstraction Layer for Portable Applications in the Domain of Wireless Sensor Networks*. In Proceedings of The 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09), Barkeley, California, USA, November 2009.

[Serna Oliver 10a] Ramon Serna Oliver & Gerhard Fohler. *Timeliness in Wireless Sensor Networks: Common Misconceptions*. In Proceedings of the 9th International Workshop on Real-Time Networks RTN'2010, Brussels, Belgium, July 2010.

[Serna Oliver 10b] Ramon Serna Oliver & Ivan Shcherbakov. *The WASP OS API*. online, `http://rts-wiki.eit.uni-kl.de/WASP/index.html`, 2009-2010.

[Serna Oliver 10c] Ramon Serna Oliver, Ivan Shcherbakov & Gerhard Fohler. *An Operating System Abstraction Layer for Portable Applications in Wireless Sensor Networks.* In Proceedings of The 25th ACM Symposium on Applied Computing (SAC 2010), Sierre, Switzerland, March 2010. ACM.

[Sharifi 06] M. Sharifi, M.A. Taleghan & A. Taherkordi. *A Middleware Layer Mechanism for QoS Support in Wireless Sensor Networks.* In Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006. International Conference on, pages 118–118, April 2006.

[Shashi Prabh 07] K. Shashi Prabh & T.F. Abdelzaher. *On Scheduling and Real-Time Capacity of Hexagonal Wireless Sensor Networks.* In Proceedings of the 19th Euromicro Conference on Real-Time Systems, ECRTS07, pages 136–145, July 2007.

[Song 09] Wen-Zhan Song, Renjie Huang, B. Shirazi & R. LaHusen. *TreeMAC: Localized TDMA MAC protocol for real-time high-data-rate sensor networks.* In Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on, pages 1–10, March 2009.

[Soyturk 08] M. Soyturk & D.T. Altilar. *Reliable real-time data acquisition for rapidly deployable mission-critical Wireless Sensor Networks.* In INFOCOM Workshops 2008, IEEE, pages 1 –6, April 2008.

[Stanley-Marbell 08] Phillip Stanley-Marbell, Twan Basten, Jerome Rousselot, Ramon Serna Oliver, Holger Karl, Marc Geilen, Rob Hoes, Gerhard Fohler & Jean-Dominique Decotignie. *System Models in Wireless Sensor Networks.* Rapport technique, Eindhoven University of Technology Department of Electrical Engineering Electronic Systems, May 2008.

[Turau 06] Volker Turau, Matthias Witt & Christoph Weyer. *Analysis of a Real Multi-hop Sensor Network Deployment: The Heathland Experiment.* In in Proc. Third International Conference on Networked Sensing Systems (INSS 2006), 2006.

[Varga 01] Andras Varga. *The OMNET++ discrete event simulation system.* In Proceedings of the European Simulation Multiconference, pages 319–324, Prague, Czech Republic, June 2001.

[Wang 09] Yunbo Wang, M.C. Vuran & S. Goddard. *Cross-Layer Analysis of the End-to-End Delay Distribution in Wireless Sensor Networks.* In Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE, pages 138–147, Dec. 2009.

[Watteyne 05] Thomas Watteyne & Isabelle Auge-Blum. *Proposition of a Hard Real-Time MAC Protocol for Wireless Sensor Networks.* In Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). IEEE Computer Society, 2005.

[Watteyne 06] Thomas Watteyne, Isabelle Augé-Blum & Stéphane Ubéda. *Dual-mode real-time MAC protocol for wireless sensor networks: a validation/simulation approach.* In InterSense '06: Proceedings of the first international conference on Integrated internet ad hoc and sensor networks, page 2, New York, NY, USA, 2006. ACM.

[Woo 03] Alec Woo, Terence Tong & David Culler. *Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks.* In In Proceedings ACM SenSys '03, pages 14–27. ACM Press, 2003.

[Yarvis 02] Mark D. Yarvis, W. Steven Conner, Lakshman Krishnamurthy, Jasmeet Chhabra & Brent Elliott. *Real-World Experiences with an Interactive Ad Hoc Sensor Network.* In ICPPW '02: Proceedings of the 2002 International Conference on Parallel Processing Workshops, page 143, Washington, DC, USA, 2002. IEEE Computer Society.

[Yu 03] Yang Yu, Bhaskar Krishnamachari & Viktor K. Prasanna. *Issues in Designing Middleware for Wireless Sensor Networks.* IEEE Network, vol. 18, pages 15–21, 2003.

[Zhao 03] Jerry Zhao & Ramesh Govindan. *Understanding packet delivery performance in dense wireless sensor networks.* In Proceedings of the 1st international conference on Embedded networked sensor systems. SenSys03, pages 1–13, New York, NY, USA, 2003. ACM.

[Zhou 04] Gang Zhou, Tian He, Sudha Krishnamurthy & John A. Stankovic. *Impact of radio irregularity on wireless sensor networks.* In MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services, pages 125–138, New York, NY, USA, 2004. ACM.

Glossary

Application Programming Interface
An Application Programming Interface (API) is an interface implemented by a software program to enable its interaction with other softwares. 17, 97, 98, 100–104

application specific integrated circuits
Application specific integrated circuits (ASIC) are non general-purpose ICs customized for a particular use. 10

Carrier Sense Multiple Access
Carrier Sense Multiple Access is a channel-access scheme in which multiple nodes access a shared medium. Before transmitting, nodes perform a carrier sense operation and proceed only if the medium is free. 6, 20, 21

Central Limit Theorem
The Central Limit Theorem is a theorem stating that the sum of a multiple R.V. following the same distribution tends to converge into a R.V. normally distributed. For a formal definition see e.g. [Bulmer 67]. 45, 52, 84

Cumulative Distribution Function
The cumulative distribution function shows the probability of an event being less or equal than a certain value. For a formal definition see e.g. [Bulmer 67]. 43, 44, 49

Earliest Deadline First
Earliest Deadline First is a scheduling strategy selecting tasks for execution based on their assigned deadline. 2, 20

Exponential Weighted Moving Average
Exponential Weighted Moving Average refers to an averaging method introducing an exponential factor ($0 \leq \alpha \leq 1$). The weight of a new sample (δ) on the exponential average ($\bar{x}$) at time t is calculated as follows: $\bar{x}_t = \alpha\delta + (1-\alpha)\bar{x}_{t-1}$. 44, 48, 50, 51

field programmable gate array
Field programmable gate array (FPGA) are ICs made of programmable logic components that can be reprogrammed after manufacturing. 7, 10

Frequency Division Multiple Access
Frequency Division Multiple Access is a channel-access scheme which divides the spectrum into non-overlapping frequency ranges. 20

Integrated Circuit
An integrated circuit is a hardware system commonly referred as a "chip". 10

Medium Access Control
Medium Access Control refers to the network layer responsible of establishing the communication between two neighbor nodes. Typical duties of this layer include carrier sensing and collision detection/avoidance. 3, 6, 17, 19–21, 66

micro controller
Micro-Controllers (μC) are hardware systems typically composed of a CPU, memory and a number of registers and I/O devices embedded into a single IC. 7, 10, 14, 15

Mobile Ad-Hoc Network
A Mobile Ad-Hoc Networks is a self-configuring network made of mobile devices connected by wireless links. 5

Operating System
The Operating System is a software (or firmware) layer providing an interface between the system hardware and the applications. 1, 9, 16–18, 31, 58–60, 96–98, 100–104, 135

Operating System Abstraction Layer
Operating System Abstraction Layer (OSAL) is an abstraction layer implemented to unify the API of different OS. 8, 17, 59, 79, 96–104, 136

Probability Density Function
The probability distribution function of a R.V. defines the probability of a certain event to happen. For a formal definition see e.g. [Bulmer 67]. 48, 49, 91

Quality of Service
Quality of Service refers to the quantitative evaluation of one or several qualitative metrics of a system. iii, 7, 9, 22–26, 31, 36, 53–61, 67, 90, 95, 135, 136, 139

Radio Frequency
Radio Frequency refers to the transmission of radio signals through the air. 6, 10, 12, 14, 79

Random Access Memory
Random Access Memory is kind of memory technology allowing access to non-sequential data fields. 12

Random Variable
Random Variable are statistical variables representing a succession of stochastic events. 43–46

Rate Monotonic
Rate Monotonic is a scheduling strategy selecting the priority of tasks based on their periodicity. 2

Read Only Memory
Read Only Memory is a kind of memory that cannot be erased or rewritten. 12

Real-Time System
Real-Time Systems are hardware or software systems in which the correctness of the results is subject to predefined timeliness constraints. 1, 2, 18, 34, 37

Hard Real-Time Systems are RTS in which the violation of their timeliness constraints could result on fatal consequences. 2, 36

Soft Real-Time System are RTS in which the violation of their timeliness constraints produces the degradation of the system performance, although they do not necessarily result in fatal consequences. 2

Time Division Multiple Access
Time Division Multiple Access is a channel-access scheme used especially in wireless communication (for example, most 2G mobile-phone standards). 6, 19–21, 28

Virtual Machine
Virtual Machines (VM) are software environments emulating the behavior of hardware systems (e.g. JAVA VM). 18, 23

Wireless Sensor Network
A Wireless Sensor Network is constituted by a number of sensor nodes communicating with an ad-hoc infrastructure running at least one common application in a collaborative manner. iii, 1, 3–7, 9, 10, 12–14, 16, 18–23, 25, 26, 29–31, 33, 35–40, 43, 51–59, 61, 62, 79, 81, 85, 88, 95, 111, 135, 139

Summary

Chapter I

This chapter presents a short introduction to the areas of research explored in this thesis. The purpose is to position the reader with respect to the organization and content of this thesis.

Chapter II

This chapter overviews the necessary background to further developing the contents of this thesis. Through its sections, it presents a prospective analysis of the state-of-the-art and exposes the main open issues and points of interest with respect to real-time and Wireless Sensor Network (WSN).

The chapter covers the architecture of typical WSNs including aspects from the software and hardware platform as well as the Operating System (OS). Further, it surveys the main aspects regarding real-time and the existing state-of-the-art with respect to the networking protocols. Following, the chapter brings up the essential terminology regarding Quality of Service (QoS) and establishing a solid meaning for terms with divergent interpretations.

The chapter concludes with an enumeration of common misconceptions and misleading practices in current real-time methods for WSN.

Chapter III

This chapter explores the implications of the commonly adopted notion of timeliness in current real-time approaches for WSNs. In contrast to the classic notion, it presents a generalized timeliness notion providing means to express meaningful timeliness in typical WSNs without restraining the feasibility of methods.

As a proof of concept, the chapter introduces a probabilistic algorithm for the estimation of end-to-end latency of message transmissions in WSNs. The estimation is based on the generalized timeliness notion obtaining the distribution function of end-to-end routing paths.

Chapter IV

The core of this chapter explores the definition and application of local and global QoS trade-offs and details a step-by-step methodology to analyze the most suitable system set-points. First, it introduces the elemental considerations defining feasible timeliness objectives enabling meaningful QoS methods. The definition of these objectives and the establishment of instruments to enable them are as important as providing means evaluating the qualities of the target system. Further, it analyzes the possibilities provided by the software and hardware platforms to exploit local trade-offs. Finally, the chapter explores the definition of global timeliness trade-offs –in new or existing real-time methods– taking into account the necessary parts having a significant impact on the real-time performance. In doing so, the chapter explores the definition of abstract models for the analysis of metrics as well as the impact of selected parameters, both from a global and local domain perspective.

Chapter V

The concepts previously introduced in this thesis are put together into a simple routing protocol as a proof of concepts. This chapter presents an approach to a simple routing protocol consisting of a modified classic tree routing protocol, which generates paths based on the generalized timeliness notion. Each hop performs the local calculations as shown in chapter III, which are used to determine the best forwarding neighbor based on a routing tree. The sink periodically broadcast control messages to reconstruct the tree and collects information regarding the end-to-end delay distributions.

Chapter VI

This chapter presents an evaluation of the main concepts introduced along the thesis. A number of scenarios evaluating different aspects are described and implemented. The evaluation is carried out by means of both simulation tools and small-scale test-bed experiments. The core of this chapter focuses on the evaluation of the routing protocol described in chapter V and provides hints for the application of the trade-off concepts presented in chapter IV.

Chapter VII

This chapter concludes the work presented in this thesis and overviews the main achievements and meaningful contributions.

Appendices

Appendix A describes in extension the Operating System Abstraction Layer (OSAL), which is already introduced in section II.2. The appendix goes in depth with details that did not fit in the main body of the thesis as well as an extensive evaluation of the overhead introduced by OSAL.

Appendix B contains additional or extended results an figures that due to their extensions or their similarity where not placed in their respective sections.

Zusammenfassung

Wireless Sensor Networks (WSN) sind dynamisch angeordnete Netzwerke, die typischerweise aus einer großen Anzahl von beliebig verteilten Sensorknoten bestehen, die alle gemeinsam mindestens eine kollektive Aufgabe erfüllen. Restriktionen wegen omnipräsenter Ressourcenknappheit und der starken Abhängigkeit von unkontrollierbaren Umwelteinflüssen sind die Hauptcharakteristiken dieses Netzwerktyps. Diese Besonderheiten führen zu schwerwiegenden Einschränkungen der Anwendbarkeit von klassischen Echtzeitmethoden, die darauf abzielen Timeliness Garantien zu geben.

Existierende Lösungen aus der Echtzeitforschung tendieren dazu Konzepte und Methoden anzuwenden, die ursprünglich nicht für diese Art von Systemen entworfen wurden. Dabei führt die Idealisierung typischer Anwendungsszenarien und das Ignorieren essentieller Einschränkungen zu einigen irreführenden Praktiken, die wiederum zu Ergebnissen mit beschränkter Validität in der realen Welt führen.

Die Entschärfung des Konflikts zwischen WSNs einerseits und den klassischen Echtzeitzielsetzungen andererseits beginnt mit einer Revision der grundlegenden Prinzipien bereits existierender Herangehensweisen. Auf diese Art und Weise präsentiert die vorliegende Dissertation einen alternativen Denkansatz, der auf einem generalisierten und den speziellen Bedürfnissen von WSNs angepassten Timeliness Verständnis basiert. Diese neue konzeptionelle Vorstellung bildet die Prämisse zur Definition von brauchbaren Echtzeitzielvorstellungen die einen neuen Bereich von Möglichkeiten eröffnen, der durch die idealisierten Vorstellungen bisher nicht abgedeckt wird.

Der Schwerpunkt dieser Dissertation besteht aus der Definition und der Anwendung von Quality of Service (QoS) Kompromissen zwischen Timeliness einerseits und anderen signifikanten QoS-Maßstäben andererseits. Die Analyse von lokalen und globalen Kompromissen definiert eine Schritt-für-Schritt-Methodik, die es erlaubt die Korrelationen zwischen diesen Qualitätsmaßstäben zu identifizieren. Diese Assoziation schafft die Voraussetzung zur Definition von alternativen Konfigurationen (set points), die die Performance des Netzwerkes zu bestimmten Zeitpunkten beeinflussen.

Nach Einführung der Grundlagen werden die oben erwähnten Konzepte als Machbarkeitsnachweis für die ausführliche Analyse in einfache Routingprotokolle eingebettet. Extensive Evaluierungen unter realistischen Bedingungen wurden sowohl in Simulationsumgebungen als auch in echten Testumgebungen durchgeführt um die Konsistenz dieses Denkansatzes zu validieren.

Kapitel I

Der Kern von Echtzeitsystemen (RTS) entwickelt sich aus den Determinismusgrundsätzen und der Berechenbarkeit der Zeit. Prozessorterminierung, Betriebssysteme (OS) und Anwendungssynchronisation sind Beispiele relevanter Probleme, die derzeit in diesem Forschungsgebiet bearbeitet werden. Andererseits verfolgen WSNs Direktkommunikation in Umgebungen mit wesentlichem Unsicherheitsfaktor. Energiesparende Kommunikation und hohe Skalierbarkeit sind die wichtigsten Faktoren, die die Forschung auf diesem Gebiet antreiben.

Das Verbinden dieser beiden grundsätzlich divergierenden Forschungsgebiete ist keine einfache Angelegenheit, da es bedeutende Versuche erfordert, die Einschränkungen und Grenzen jedes Gebiets sowie ihr Anpassungsvermögen zu verstehen. Dieses Kapitel stellt die Grundlagen anhand eines kurzen Überblicks über die Grundbegriffe zu jedem Gebiet vor.

Das Kapitel gliedert sich wie folgt: Teil I.1 stellt die Grundlagen von RTS, und die zugehörigen Begriffe vor. Teil I.2 folgt mit einer kurzen Einführung der Eigenschaften und grundsätzlichen Betrachtungen von WSNs. Am Ende des Kapitels gibt Teil I.3 einen kurzen Abriss über den Aufbau dieser Doktorarbeit.

Kapitel II

Bevor man sich den wichtigen Errungenschaften in einem Forschungsgebiet zuwendet, ist es erforderlich, den aktuellen Stand der Technik zu verstehen. Daher stellt dieses Kapitel anhand von maßgeblichen wissenschaftlichen Veröffentlichungen und zugehörigen Quellen einen Überblick über den Stand der Technik auf dem Gebiet der WSNs vor. Das Kapitel gliedert sich wie folgt:

Teil II.1 stellt kurz die Architektur von Sensorknoten und ihre einzelnen Bauteile vor. Ziel ist es, die Hauptquellen des Energieverbrauchs eines typischen Sensorknotens zu identifizieren.

Teil II.2 untersucht, wie die Bauteile und Ressourcen durch das OS verwaltet werden. Er gibt einen allgemeinen Überblick über bestehende OSs und stellt die Grundlagen einer theoretischen Abstraktionsschicht für das OS vor.

Nachdem die Hardware- und Firmware-Aspekte des Sensorknotens abgehandelt wurden, untersucht Teil II.3 die Echtzeitaspekte des aktuellen Stands der Technik. Dieser Part ist in zwei Teile aufgeteilt, die sich intensiv jeweils mit Echtzeitkommunikationsprotokollen (Teil II.3.1) bzw. Echtzeit-Frameworks und -Middleware (Teil II.3.2) beschäftigen. Aus praktischen Gründen und für ein besseres Verständnis der weiteren Ausführungen dieser Arbeit wurde das Schwerpunkt auf Echtzeit- und timelinesssbezogene Protokolle gelegt.

Der Zweck von Teil II.4 ist es, die grundlegenden Begriffe bezüglich QoS einzuführen. Insbesondere werden hier grundlegende Definitionen und Terminologien entwickelt, die in dieser Arbeit verwendet werden. Dadurch wird die Möglichkeit von Verwechslungen mit Begriffen vermindert, die häufig Gegenstand unterschiedlicher Interpretationen sind.

WSNs bilden ein aufstrebendes Gebiet der Forschung, das Eigenschaften aus vielen anderen Bereichen verbindet bzw. sich diese mit ihnen teilt. Allerdings gibt es eine große Zahl von inhärenten Eigenschaften, die große Herausforderungen darstellen und denen individuell begegnet werden muss. Aus diesem Grund untersucht Teil II.5 die wichtigsten Eigenschaften und identifiziert gemeinsame irreführenden Annahmen sowie die damit verbundenen Konsequenzen. Schließlich endet das Kapitel mit einer Zusammenfassung.

Kapitel III

Die inhärenten Eigenschaften der in Kapitel II eingeführten WSNs bilden ein ungünstiges Umfeld für die Durchsetzung der Echtzeit-Bedingungen. Die Unmöglichkeit mit Hilfe eines typischen Netzwerkstacks zeitgebundene Punkt-zu-Punkt-Übertragungen zu gewährleisten, kollidiert jedoch mit dem erstrebten Ziel durchgehend exakter Verzögerungen.

Grundlegende Aspekte legen nahe, dass alternative Ansätze untersucht werden müssen, wenn besondere Timeliness garantiert werden muss. Diese Ansätze erfordern eine wesentliche Änderung der Ziele sowie die Etablierung geeigneter Annahmen für typische WSNs.

Dieses Kapitel bietet eine eingehende Analyse der wesentlichen Fehler in den Gleichzeitigkeitsansätzen, die zurzeit in WSNs genutzt werden. Darüber hinaus erforscht sie im Detail einen neuen Weg, sich garantierter Timeliness zu nähern in Übereinstimmung mit den auferlegten Eigenschaften und Einschränkungen eines typischen WSN. Das Kapitel ist wie folgt organisiert:

Bevor sich das Kapitel den Einzelheiten widmet, regelt Teil III.1 die Terminologie sowie grundlegende Definitionen, die in der weiteren Arbeit verwendet werden.

Abschnitt III.2 ist in zwei Teile gegliedert. Zunächst wird der sehr erweiterte klassische Begriff der Gleichzeitigkeit im Bereich der WSN analysiert, indem die Grenzen und grundlegenden Mängel identifiziert werden. Ausßerdem stellt er eine allgemeine Darstellung von Gleichzeitigkeit vor, mit deren Hilfe besondere Gleichzeitigkeit in typischen WSNs ausgedrückt werden kann, ohne ihre Anwendbarkeit zu vernachlässigen. Der allgemeine Begriff unterscheidet sich von dem klassischen in Bezug auf die Anforderungen und Annahmen. Des Weiteren eröffnet er einen alternativen Weg, um Timeliness zu garantieren.

Als Beweis der Machbarkeit des Konzepts präsentiert Abschnitt III.3 einen wahrscheinlichkeitstheoretischen Algorithmus zur Abschätzung der durchgehenden Verzögerung bei Nachrichtenübertragungen in WSNs. Diese Methode basiert auf der allgemeinen Darstellung der Timeliness und ermöglicht die Verteilungsfunktion der durchgehenden Routing-Pfade mit zufriedenstellender Genauigkeit vorauszusagen. Schließlich endet Kapitel III mit einer Zusammenfassung.

Kapitel IV

Das ultimative Ziel einer Echtzeit-Methode für WSNs ist rechtzeitiges Antworten auf ausgewählte Ereignisse und gleichzeitig die Erfüllung vorher definierter QoS-Anforderungen zu gewährleisten. Doch die allgemeinen Ziele zu definieren, in denen diese Anforderungen festgelegt werden, ist keine triviale Aufgabe. Der in Kapitel III vorgestellte allgemeine Begriff der Timeliness eröffnet ein neues Spektrum an Möglichkeiten, QoS Trade-offs als ein Mittel zur Erfüllung der Timelinessseinschränungen zu erforschen, während die Spezifikation bereits realisierbarer Ziele vereinfacht wird.

Die neue Vorstellung, die auf der Verteilung durchgängiger Verzögerungen basiert, ermöglicht eine flexible Definition der Anforderungen lokaler und globaler Laufzeit-Anpassungen, was zur Verbesserung der Pünktlichkeit in bestimmten Momenten beiträgt. Diese Anpassungen können die Verfolgung realistischer durchgängiger Timeliness zulassen, die auf adaptiven QoS-Anforderungen basiert. Allerdings müssen geeignete Mechanismen sowie Interfaces, die QoS-Trade-offs ermöglichen, entwickelt werden, um die notwendigen Anpassungen durchzusetzen. Die Spezifikation der Zielvorstellungen von Timeliness in einem WSN, das den potentiellen Konflikt zwischen realisierbaren Anforderungen und die Einschränkungen durch Protokolle und Szenarien überwindet, ist nicht trivial und erfordert nach wie vor ein gewisses Maß an individueller Analyse der jeweiligen Anwendung. Insbesondere führt das Abwägen von Trade-offs zwischen Gleichzeitigkeit und anderen QoS-Metriken zu einer Reihe von potentiellen Anpassungen, die Echtzeit-Methoden ausnutzen und dabei zufriedenstellende QoS-Ebenen erreichen.

Das Kapitel ist wie folgt gegliedert:

Teil IV.1 stellt die elementaren Überlegungen vor, welche die Zielvorstellung von Timeliness definieren und aussagekräftige QoS-Methoden ermöglichen. Die Definition dieser Ziele und die Entwicklung von Instrumenten, die dies ermöglichen ist ebenso wichtig wie das Bereitstellen von Mitteln, um die Qualität des Zielsystems zu bewerten.

Abschnitt IV.2 analysiert die Möglichkeiten, die Software- und Hardware-Plattformen bereitstellen, um lokale Trade-offs zu nutzen.

Abschnitt IV.3 erforscht die Definition von allgemeinen Trade-offs zur Timeliness in neuen bzw. bestehenden Echtzeit-Methoden unter Berücksichtigung der Teile, die erhebliche Auswirkungen auf die Echtzeit-Performance haben. Die Analyse konzentriert sich auf die Aspekte, die die Anwendbarkeit der allgemeinen Vorstellung von Timeliness und die Ausnutzung von QoS-Trade-offs beeinflussen. Dabei untersucht das Kapitel die Definition von abstrakten Modellen für die Analyse von Kennzahlen sowie die Auswirkungen der gewählten Parameter, sowohl aus allgemeiner wie auch aus lokaler Domain-Perspektive.

Schließlich endet das Kapitel mit einer Zusammenfassung.

Kapitel V

Die Anwendung des verallgemeinerten Begriffs von Gleichzeitigkeit zur Laufzeit als auch die Schätzung der durchgängigen Verzögerungsverteilung benötigt Unterstützung aus dem Netzwerkverbund. Bestehende Protokolle können einfach angepasst werden, um die

Pünktlichkeitsleistung mittels des in Kapitel III vorgestellten verallgemeinerten Begriffs auszudrücken.

Ebenso kann die Schätzung der durchgängigen Verzögerungsverteilung in die Routing-Protokolle eingebettet werden und bietet Laufzeit-Schätzungen der Pfadverteilungverzögerungen.

Dieses Kapitel präsentiert ein einfaches Routing-Protokoll, bestehend aus einem modifizierten klassischen Baum-Routing-Protokoll. Dieses generiert Wege, die auf dem allgemeinen Begriff der Timeliness basieren.

Jeder Sprung führt die lokalen Berechnungen wie in Teil III.3 gezeigt durch, die verwendet werden, um die besten weiterleitenden Nachbarn zu bestimmen, die auf einem Routing-Baum basieren. Die Senke sendet in regelmäßigen Abständen Kontrollnachrichten, um den Baum zu rekonstruieren und sammeln Informationen über die durchgängigen Verzögerungsverteilungen.

Der Zweck dieses Kapitels ist es nicht, ein vollständiges Protokoll über alle wichtigen Routing-Aspekte zu erarbeiten, sondern die in dieser Arbeit entwickelten Konzepte zur Demonstration in ein einfaches Protokoll zu integrieren. Anhand der gleichen Ideen, die in diesem Kapitel vorgestellt werden, kann die gleiche Anpassung an komplexere Protokolle durchgeführt werden.

Der Inhalt dieses Kapitels ist wie folgt gegliedert:

Teil V.1 gibt einen Überblick über die Grundlagen von Routing-Protokollen, die auf Routing-Bäumen basieren.

Teil V.2 beschreibt das sich der Timeliness bewusste Routing-Protokoll (TARP), in das die Schätzung der durchgängigen Verzögerungsverteilung eingebettet ist. Es wird detailliert die Menge der Ressourcen aufgelistet, die das Protokoll benötigt.

Eine einfache Erweiterung des Protokolls, die um Multi-Path- und Multi-Sink-Szenarien erweitert wurde, wird in V.3 vorgestellt.

Kapitel V endet schließlich mit einer Zusammenfassung.

Kapitel VI

Dieses Kapitel stellt eine Bewertung der wichtigsten Konzepte vor, die in dieser Arbeit eingeführt werden. Eine Reihe von Szenarien, welche die verschiedenen Aspekte bewerten, wird beschrieben und implementiert. Die Auswertung erfolgt sowohl mit Hilfe von Simulationswerkzeugen als auch anhand kleiner Experimente in Testumgebungen. Der Kern dieses Kapitel konzentriert sich auf die Auswertung des Routing-Protokolls aus Kapitel V und liefert Hinweise für die Anwendung der Trade-off-Konzepte, die in Kapitel IV vorgestellt werden.

Das Kapitel ist wie folgt gegliedert:

Teil VI.1 gibt einen Überblick über die Bewertungsumgebungen und die Szenarien, die als Grundlage für die Evaluierungsprozesse dienen. Diese Szenarien sind so konzipiert, dass sie die Eigenschaften, die von realen Einsätzen sowie kleinen Set-ups extrahiert werden, berücksichtigen, um spezifische Netzwerk-Aspekte zu validieren. Der Abschnitt stellt die eigentliche Testumgebungsplattform sowie die Simulationsumgebung vor, in denen die Bewertungsexperimente durchgeführt wurden. Eine Beschreibung der beiden

Plattformen, einschließlich der Implementierungs- und Konfigurationseinzelheiten, wird ebenfalls vorgestellt.

Die Ergebnisse, die im Auswertungsprozess gewonnen wurden, werden in Teil VI.2 zusammengefasst. Der Abschnitt ist in zwei Teile gegliedert, entsprechend der Ergebnisse aus den Simulationsläufen als auch entsprechend der Zahlen aus einer Reihe von kontrollierten Testumgebungsexperimenten.

In Teil VI.3 schließt das Bewertungskapitel mit einer Diskussion über die vorgestellten Ergebnisse. Die Ergebnisse und Zahlen aus Teil VI.2.2 werden kommentiert und mit den Simulationen aus Teil VI.2.1 verglichen.

Kapitel VI endet schließlich mit einer Zusammenfassung.

Kapitel VII

Das Kapitel kompletiert diese Arbeit mit einer Übersicht über die wichtigsten Fortschritte und bedeutendsten Beiträge.

Appendix A

Die Einrichtung von Mechanismen für eine ordnungsgemäße Kontrolle der Hardware-Plattform in die Software-Frameworks wirft eine Reihe von Fragen zur Portabilität auf. Der Operating System Abstraction Layer (OSAL) wurde entwickelt, um diese Probleme anzugehen und Konflikte zwischen verschiedenen Software-und Hardware-Plattformen zu vermindern. Insbesondere spricht sie die Diskrepanzen an zwischen den verschiedenen OS hinsichtlich ihrer funktionellen Application Programming Interfaces (API), Hardware-Konfigurationsmechanismen, Ressourcenmanagement und Handhabung von Peripheriegeräten.

Appendix B

Dieser Anhang enthält zusätzliche oder erweiterte Ergebnisse und Zahlen, die aufgrund ihrer Erweiterungen oder ihrer Gleichartigkeit nicht in den jeweiligen Abschnitten erwähnt wurden.