Vasco Nuno Sousa Simões Pereira

Performance Measurement in Wireless Sensor Networks

Medição de Desempenho em Redes de Sensores Sem Fios

Tese de doutoramento do Programa de Doutoramento em Ciências e Tecnologias da Informação,
orientada por Prof. Doutor Edmundo Monteiro e Prof. Doutor Jorge Sá Silva
e apresentada no Departamento de Engenharia Informática
da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

2016

· U C ·

UNIVERSIDADE DE COIMBRA

Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Informática

Medição de Desempenho em Redes de Sensores Sem Fios

Vasco Nuno Sousa Simões Pereira

Programa de Doutoramento em Ciências e Tecnologias da Informação
Tese de Doutoramento submetida à Universidade de Coimbra

Fevereiro de 2016

University of Coimbra

Faculty of Sciences and Technology

Department of Informatics Engineering

Performance Measurement in Wireless Sensor Networks

Vasco Nuno Sousa Simões Pereira

Doctoral Program in Information Science and Technology

PhD Thesis submitted to the University of Coimbra

February 2016

Under supervision of

Professor Doutor Edmundo Heitor da Silva Monteiro

Professor Catedrático do Departamento de Engenharia Informática
da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Professor Doutor Jorge Sá Silva

Professor Auxiliar do Departamento de Engenharia Informática
da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Agradecimentos (Acknowledgements)

Durante o desenvolvimento deste trabalho várias foram as pessoas e instituições que de alguma forma permitiram que ele fosse possível e a quem quero expressar os meus agradecimentos.

Gostaria de agradecer em primeiro lugar ao Dr. Edmundo Monteiro e ao Dr. Jorge Sá Silva, meus orientadores, todo o apoio prestado ao longo da elaboração desta tese. A sua ajuda na definição do trabalho, nos comentários e revisões de textos e artigos, e também nas discussões que proporcionaram, foi imprescindível.

Ao projecto GINSENG, pela possibilidade que me deu de colaborar num projecto na área das redes de sensores sem fios de desempenho controlado, em ambientes industriais. O trabalho desenvolvido e os conhecimentos adquiridos foram essenciais na definição e posterior desenvolvimento do trabalho da tese.

Aos meu colegas do Grupo de Sensores, Ricardo, André, Jorge e Dien pelo debate de ideias que proporcionaram.

À Fundação Calouste Gulbenkian e ao Prof. Thiemo Voigt a oportunidade que me deram de ter estado em Estocolmo no SICS, onde pude aprofundar os meus conhecimentos de Contiki.

Ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, em especial ao Dr. Paulo de Carvalho e Dr. António Mendes como presidentes, as facilidades que me proporcionou para que eu tivesse tempo para concluir este trabalho.

Foreword

The work described in this thesis was done at the *Laboratory of Communications and Telematics* (LCT) of the *Centre for Informatics and Systems of the University of Coimbra* (CISUC). The research work was partially done in the context of project GINSENG.

GINSENG – Performance Control in Wireless Sensor Networks, was a project funded by the European Union, under the Seventh Framework Programme (FP7), from September 2008 to February 2012. The project developed a novel Wireless Sensor Network with controlled performance, targeted on industrial monitoring and control applications. The main achievement of the project was to design, deploy and evaluate a complete end-to-end system for performance control using Wireless Sensor Networks.

The author also participated as expert for the International Organization for Standardization (ISO), specifically in the ISO/IEC JTC 1 study group on Sensor Networks (International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC) - JTC 1 SGSN Study Group on Sensor Networks - Oslo meeting, June 2009.

The work done during this thesis resulted in the following journal articles, conference papers and technical reports:

JOURNALS

- Luiz H.A. Correia, Thanh-Dien Tran, **Vasco Pereira**, João C. Giacomin, Jorge M. Sá Silva, "DynMAC: A resistant MAC protocol to coexistence in wireless sensor networks", Elsevier Computer Networks, Volume 76, 15 January 2015, Pages 1–16 (http://dx.doi.org/10.1016/j.comnet.2014.10.019), Impact Factor: 1.282, 5-Year Impact Factor: 1.871.

- **Vasco Pereira**, J. Sá Silva, A. Cardoso, P. Gil, P. Furtado, R.M. Silva, J. Cecílio, A. Santos, A. Gomes, E. Monteiro, J. Ó and R. Eiras, "Redes de Sensores sem Fios com Desempenho Controlado – Projecto FP7 GINSENG", Ingenium (Revista da Ordem dos Engenheiros), 2011.

CONFERENCE PAPERS

- **Vasco Pereira**, Edmundo Monteiro, Jorge Sá Silva, "A data fusion protocol for WSN performance and data retrieval", in Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS), 2016, to be published.

- T.D. Tran, J. Oliveira, J. Sá Silva, **V. Pereira**, N. Sousa, D. Raposo, and F. Cardoso, "A Scalable Localization System for Critical Controlled Wireless Sensor Networks," in Proc. of the ICUMT 2014, 6th International Congress on Ultra Modern Telecommunications and Control Systems, 2014.

- **Vasco Pereira**, Jorge Sá Silva and Edmundo Monteiro, "A framework for Wireless Sensor Networks performance monitoring", in Proc. of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012.

- **Vasco Pereira**, Jorge Silva, Jorge Granjal, Ricardo Silva, Edmundo Monteiro, Qiang Pan, "A Taxonomy of WSN with QoS", in Proc. of 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2011.

PARTICIPATION IN TECHNICAL REPORTS

- "D1.3 - Final GINSENG Architecture, Scenarios and Quality of Service Measures", FP7 GINSENG (INFSO-ICT-224282) Deliverable D1.3, WP1, October 2010.

- "D5.2 - Periodic Work Package Report: Dissemination, exploitation and contribution to standards", FP7 GINSENG (INFSO-ICT-224282) Deliverable D5.2, WP5, March 2010.

- "D1.1 - GINSENG Architecture, Scenarios and Quality of Service Measures", FP7 GINSENG (INFSO-ICT-224282) Deliverable D1.1, WP1, November 2009.

- "D5.1 - Periodic Work Package Report: Dissemination, exploitation and contribution to standards", FP7 GINSENG (INFSO-ICT-224282) Deliverable D5.1, WP5, June 2009.

- "Technical Document of ISO/IEC JTC 1 - Study on Sensor Networks (Version 3)." Study Group on Sensor Networks, September 2009.

Abstract

The use of *Wireless Sensor Networks* (WSNs) has opened the possibility of a new set of applications that are having a growing impact in personal and business activities. However, most of its application scenarios have been restricted to non-critical environments, with WSN being operated with no controlled performance. The aim to extend the flexibility and unique characteristics of these networks to a broader set of applications and scenarios, such as industrial and health care, poses new challenges that must be met with a new approach. In such environments, WSNs may have significant benefits over traditional networks, such as enabling a deeper control, lowering deployment and maintenance costs, and by offering simple reconfiguration and adaptation to changing business models.

To enable WSNs with controlled performance the first step is to be able to characterize and describe the requirements that the network needs to fulfil. The second step is to be able to translate those requirements into effective metrics. The metrics to be used must be adapted to the unique characteristics of WSNs, taking into account its processing, energy and storage restrictions. The next step is to monitor those metrics, and allow for their debugging when necessary, a procedure that involves their collection to a central base station where further treatment, with better resources, is possible and where an effective network monitoring can be achieved. The last step completes the cycle and corresponds to the ability to dynamically act in the network, based on the metrics received, either automatically (by each node or by a central monitoring tool connected to the sink) or through the Network Manager.

In this thesis, the performance control life cycle of a WSN is addressed, especially considering the performance needed in industrial facilities, one of the most demanding scenarios for these networks, requiring not only strict performance boundaries but also real-time monitoring of the network. Valuable insights of these environments were possible through the participation in project FP7 GINSENG. First, a new classification of WSN application scenarios, that also includes critical environments, and a proposal of a new taxonomic tree of WSNs *Quality of Sensing* (QoSensing) requirements, including WSNs with performance control, is presented. The objective is to characterize WSNs needs, both in the information as in the network planes, and to create a reference classification that lays the foundations for the creation of effective metrics that permit the evaluation and verification of each requirement. The taxonomy was applied to different types of GINSENG scenarios and also to well-known types of applications found in the literature for validation. Having as reference the taxonomy created, and the specificities of WSNs, a study of different types of metrics is presented and their characteristics and applicability to WSNs discussed. In this context, collective metrics are introduced as a useful type of metric to address the evaluation of the global network QoSensing, while using least resources than other types of metrics and hiding the normal fluctuation of values in networks subject to many hazards. Simulations showed that collective metrics

are an efficient alternative to individual or aggregated metrics, in the assessing of the global QoSensing of a WSN. Next, the Network performance branch of the previously proposed taxonomy is analysed and a general set of metrics, adapted to each of the phases of the life cycle of a WSN, is proposed to address it. A reduced set of metrics specifically targeted to WSNs in industrial environments, focusing collective metrics for the operation phase of the network, is also proposed. The control and maintenance of levels of performance, based on a continued evaluation of specific metrics and in the dynamic actuation in the network was also addressed, with the participation in the creation of a new protocol that deals with interferences. Finally, a new protocol that collects performance data from the network is proposed. By using data fusion, the protocol presents an effective way to monitor the global performance of the network, while guaranteeing that if some error or problem occurs, an alert is generated and immediately sent to sink. The evaluation of this protocol, made by simulation, showed a decrease in the energy spent and in the interference generated by the number of packets sent, while providing for a global knowledge of the overall performance of the network. The thesis also contributed to project GINSENG, namely in the classification of the project scenarios, according to the taxonomy proposed, and in the specification of the performance metrics to be used.

Keywords: monitoring, performance evaluation, performance metrics, quality of service, taxonomy, wireless sensor networks.

Resumo

A utilização de *Redes de Sensores sem Fios* (RSSF) possibilitou o aparecimento de um novo conjunto de aplicações com um impacto crescente em vários ramos de actividade, desde pessoais a negócios. No entanto, a maioria dos seus cenários de utilização tem estado restrita a ambientes não críticos, com as RSSF a funcionarem sem controlo de performance. O objectivo de estender a flexibilidade e demais características únicas destas redes a um conjunto mais vasto de aplicações e cenários, como cenários industriais ou de assistência médica, cria novos desafios que só podem ser superados através de uma nova abordagem. Nestes cenários, as RSSF apresentam benefícios significativos em relação às redes tradicionais, como um controlo mais detalhado do meio, custos de implementação e manutenção mais baixos, e por oferecerem uma reconfiguração simples e adaptável à evolução dos modelos de negócio.

De modo a permitir a existência de RSSF com performance controlada, o primeiro passo é conseguir caracterizar e descrever os seus requisitos. O segundo passo é traduzir esses requisitos em métricas. As métricas a usar necessitam de estar adaptadas às características únicas das RSSF, tendo em conta as suas restrições de processamento, de energia e de memória. De seguida, é necessário proceder à monitorização dessas métricas, e permitir operações de depuração de erros na rede, o que envolve o envio dessas mesmas métricas para uma estação base central, onde podem ser sujeitas a uma análise mais profunda através da utilização de melhores recursos, e onde uma monitorização efectiva da sua performance pode ser efectuada. O último passo completa o ciclo e corresponde à capacidade de actuar dinamicamente na rede, com base em valores obtidos através de métricas, quer de forma automática (através do próprio nó ou através de uma ferramenta de monitorização ligada à estação base), quer através da actuação do gestor de rede.

Nesta tese, o ciclo de vida de uma RSSF com performance controlada é abordado, sendo especialmente focadas as redes para ambientes industriais, um dos cenários mais complexos e exigentes para as RSSF, requerendo não só limites apertados para a performance, como também uma monitorização em tempo-real. Os conhecimentos adquiridos pela participação no projecto FP7 GINSENG foram especialmente importantes para a análise deste cenário. Primeiro, uma nova classificação dos cenários usados pelas aplicações de RSSF (com e sem performance controlada) e uma nova taxonomia de requisitos para RSSF, são propostas. O objectivo é caracterizar os requisitos necessários às RSSF, quer no plano da informação, como no de rede, e criar uma classificação de referência que permita o desenvolvimento posterior de métricas que possam avaliar cada um desses requisitos. A taxonomia foi aplicada a vários cenários do projecto GINSENG e também em aplicações típicas de RSSF, para validação. Tendo como base a taxonomia desenvolvida e as especificidades das RSSF, é efectuado depois um estudo sobre os diferentes tipos de métricas, os quais são depois analisados tendo em conta a sua aplicabilidade. Neste contexto são introduzidas as métricas colectivas. Este tipo de métricas é adequado à medição do desempenho global das RSSF, gastando menos

recursos que os demais tipos e escondendo as normais flutuações de valores produzidos por estas redes. Testes por simulação mostraram que as métricas colectivas são uma alternativa eficiente quer às métricas individuais como às agregadas, na avaliação do desempenho global da RSSF (ou a sua qualidade sensorial). De seguida, o ramo de performance de rede da taxonomia proposta é analisado e como resultado é criado um quadro de referência com métricas para o poder avaliar, divididas pela fase de vida da rede em que devem ser aplicadas. A proposta de um quadro de métricas mais reduzido, contendo métricas especialmente adequadas à avaliação de desempenho de RSSF em ambientes industrias e que foca o uso de métricas colectivas para a fase de operação da rede, é também proposto. O controlo e manutenção do desempenho da rede baseado numa contínua avaliação de métricas específicas é também abordado através da participação na criação de um novo protocolo que lida com interferências na rede. Por fim, é proposto um novo protocolo para recolha das informações de performance da rede. Este protocolo permite a avaliação global de desempenho da rede ao mesmo tempo que garante que, caso necessário, um alerta seja gerado e enviado directamente para a estação base. A avaliação deste protocolo foi feita através de simulação, tendo mostrado ganhos de energia e redução do número de interferências na rede, quando comparado com o envio de métricas individuais, mantendo um conhecimento constante do desempenho global da rede. Esta tese também contribuiu para o projecto GINSENG, nomeadamente na classificação dos seus vários cenários, de acordo com a taxonomia desenvolvida, e na especificação das métricas de performance a usar.

Palavras-chave: avaliação de desempenho, métricas de desempenho, monitorização, redes de sensores sem fios, taxonomia, qualidade de serviço.

I Table of Contents

II List of Figures

LIST OF FIGURES

III List of Tables

IV Acronyms

6LoWPAN	*IPv6 over Low power Wireless Personal Area Networks*
ACK	*Acknowledge*
ADC	*Analog-to-Digital Converter*
ADSL	*Asymmetric Digital Subscriber Line*
BTC	*Bulk Transport Capacity*
CCA	*Clear Channel Assessment*
CIA	*Confidentiality, Integrity and Availability*
CISUC	*Centre for Informatics and Systems of the University of Coimbra*
CSMA	*Carrier Sense Multiple Access*
DiMo	*Distributed Node Monitoring in Wireless Sensor Networks*
DSN	*Deployment-Support Network*
DynMAC	*Dynamic MAC*
FCAPS	*Fault, Configuration, Accounting, Performance and Security*
FCTUC	*Faculty of Sciences and Technology of the University of Coimbra*
FP7	*Seventh Framework Programme*
GID	*Group Identification*
GinMAC	*GINSENG Medium Access Control*
ICMP	*Internet Control Message Protocol*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
IPPM	*Internet Protocol Performance Metrics*
ISA	*International Society of Automation*
ISM	*Industrial, Scientific and Medical*
ISO	*International Organization for Standardization*
ITU	*International Telecommunication Union*
LCT	*Laboratory of Communications and Telematics*
LQI	*Link Quality Indication*
MAC	*Medium Access Control*
MIB	*Management Information Base*
MP	*Measurement Points*
OWAMP	*One-Way Active Measurement Protocol*
PAD	*Passive Diagnosis for Wireless Sensor Networks*
PER	*Packet Error Ratio*
QoE	*Quality of Experience*
QoS	*Quality of Service*
QoSensing	*Quality of Sensing*
RAM	*Random Access Memory*
RDC	*Radio-duty Cycle*
RSSF	*Redes de Sensores Sem Fios*

RSSI	*Received Signal Strength Indicator*
SGSN	*Study Group on Sensor Networks*
SLA	*Service Level Agreement*
SNIF	*Sensor Network Inspection Framework*
SNMP	*Simple Network Management Protocol*
SNMS	*Sensor Network Management System*
SNR	*Signal to Noise Ratio*
StDev	Standard deviation
TAG	*Tiny AGgregation*
TCP	*Transmission Control Protocol*
TDMA	*Time Division Multiple Access*
TTL	*Time to Live*
TWAMP	*Two-Way Active Measurement Protocol*
UDGM	*Unit Disk Graph Medium*
WLAN	*Wireless Local Area Network*
WMCAP	WSNs Metrics Collection and Alert Protocol
WSAN	*Wireless Sensor and Actuator Networks*
WSN	*Wireless Sensor Network*

Chapter

1

Introduction

This thesis describes the research by the author on the subject of performance measurement in WSNs and was accomplished at the *Laboratory of Communications and Telematics* (LCT) of the *Centre for Informatics and Systems of the University of Coimbra* (CISUC). This chapter gives an overview of the context of the thesis, the motivation for the work, describes the objectives and contributions, and presents the structure of the thesis.

1.1 MOTIVATION

From washing machines to cell-phones, every device around living and working spaces is gaining computation and control capabilities, a tendency that will soon include even dispensable goods like groceries. In an early future, the concept of "Ambient Intelligence", where multiple devices gather and process information from many different sources in order to control physical processes and interact with humans, will become real. Although embedded computing is not a new concept, it is the recent low-cost integration of computing, communication, and sensing, that has turned *Wireless Sensor and Actuator networks* (WSANs), hereafter referred simply as *Wireless Sensor Networks* (WSNs) an active area of research in the last decade. The goal is to make these new technologies reliable and unobtrusive, while providing for a better control of the real world [1].

A WSN is built from spatially distributed autonomous devices. Each device (sensor and/or actuator, commonly referred only as sensor) has computation, wireless communication, and sensing or controlling capabilities. These devices can be deployed randomly in uncontrolled environments, such as in disaster or hostile areas, for environmental control and surveillance, or in a planned manner, as in industrial environments. In addition, nodes can be mobile themselves, changing position after initial deployment, as to compensate for erroneous initial deployment, or to obtain other data. Together, they constitute a network that gathers and distributes the information obtained by individual devices, enabling collaborative tasks where no node by itself can fulfil all the requirements. Wireless communications are a key factor for this new concept of network. Not only wires are more expensive and hard to maintain but also prevent mobility. Furthermore, it is possible to put sensors and actuators closer to the phenomenon that they are supposed to control or monitor, and to easily integrate new ones into the network after the initial deployment. Integrating all the referred capabilities into small-scale low–cost devices, will not only improve many existing applications, but also opens the door to a new world of applications, assuming that their intrinsic limitations, as well as their strengths, are well understood. Some of the foreseeable applications are disaster relief applications, environment control and biodiversity mapping, intelligent buildings, facility management, machine surveillance and preventive maintenance, precision agriculture, health care, logistics, industrial plants and battlefield monitoring and command.

However, WSNs flexibility is either an advantage as it is a problem. Different deployment and maintenance options, variable energy supply needs, and different application requirements, create a complex network where there is no unique optimal solution, and where trade-offs must be exploited. Also, a balance must be established between common needs of general applications, and specific needs of specific applications. As an example, most applications will require multi-hop wireless communication mechanisms, but only some will require strict packet delivery timing.

Furthermore, WSNs flexibility will require innovative mechanisms for the communication network, as well as new architecture and protocol concepts. Some of the mechanisms that a typical WSN will demand are multi-hop wireless communication, energy-efficient operation, auto-configuration, collaboration and in-network processing, data-centric communications (as opposition of traditional address-centric networks) and locality (each node should only relate to its immediate neighbours). Furthermore, when applying these networks to critical scenarios, the provided performance must be carefully monitored, to assure that the initial network demands are still being verified. If possible, this monitoring should happen in real-time.

The new possibilities that WSNs bring are visible not only in the increasing amount of research being done in the WSN area, but also in the growing number of companies offering commercial WSN solutions. In fact, research and commercial interest in the area of WSNs is currently growing exponentially, which is manifested in the number of web pages that address the theme (Google: 10,700,000 hits for sensor networks and 7,360,000 hits for wireless sensor networks in February 2015, that compares to 1,180,000 hits for sensor networks and 3,110,000 for wireless sensor networks in April 2008 [2]).

Typically, a WSN deployment scenario features an uncontrolled environment, with random placement of nodes, relying on the self-configuration of nodes, with high levels of redundancy to achieve robustness and does not provide any performance assurances. Thus, typical WSNs are not prepared to run performance critical applications that demand *Quality of Service* (QoS). However, a controlled performance scenario is important in many areas such as industrial plants and health monitoring. In such environments, WSNs may have significant benefits over existing networks, such as enabling a deeper control, lowering deployment and maintenance costs, and by offering simple reconfiguration and adaptation to changing business models. In this scenario, a careful deployment plan, where sensors/actuators are distributed orderly in a controlled environment, and application-specific performance targets, are essential.

Despite being an essential requirement for many WSNs scenarios, performance control in WSNs is yet a challenge to be met. To accomplish this objective, QoS, a broad term used to describe the overall experience a user or application will receive over a network, must be monitored and controlled. While QoS in traditional networks has been extensively researched, the same did not happen in WSNs. The reason is that WSNs are very different from traditional networks, raising the definition and measurement of QoS to a higher complexity level. In traditional networks, QoS refers to the assurance that the network can provide in an end-to-end basis. The QoS requirements of users connected to a data network are a statement of the level of quality the users expect to get from the applications they run or from the services they subscribe. Some of the most important QoS parameters that can be measured and monitored are network availability, bandwidth, packet delay, jitter and packet loss. However, these parameters are not enough to fulfil WSNs needs. Also, WSNs have several limitations like processing power, energy,

storage, communication capabilities, bandwidth, dynamic topology, non-uniform traffic, scalability, or multiple sinks, what makes the QoS support in such networks even much more challenging [3][4]. Furthermore, WSNs QoS requirements depend heavily on the application that is used. While some applications may require real-time data, as video-surveillance applications where actions must be taken in real-time, others may only require minimum space coverage and lifetime, as in temperature monitoring.

Enabling the expansion of WSNs flexibility and reduced costs to new scenarios, where performance is essential, will open these networks to an all-new scope of applications, and is the main motivation of the work presented in this thesis. To make it possible, WSNs performance characteristics must be studied, new types of metrics proposed and an efficient method of performance data collection created. Together, they will be an important step to achieve WSNs with controlled performance.

1.2 THESIS CONTEXT

The initial part of the thesis was done under project GINSENG – Performance Control in Wireless Sensor Networks [5]. This thesis main contribution was made in Task 1.2 – Measure of Performance. The aim of GINSENG was to develop a novel WSN with performance control to be used in industrial environments. The WSN proposed met application-specific performance targets, integrated with industry resource management systems and was tested in a real industrial setting where performance was critical. Besides the specific work that was made for GINSENG, the context provided by this project was also used as an example of a demanding network, with performance control needs, and used along the rest of the thesis work.

1.3 OBJECTIVES AND CONTRIBUTIONS

After establishing the motivation and the context of the thesis, specific Research Questions were elaborated that define and systematize the problem to address.

Research Question 1: Which are the specific performance requirements of WSNs?

Research Question 2: How to measure the performance of WSNs, while considering all the limitations of these networks?

Research Question 3: How to collect performance data from the network while keeping a low overhead that does not, by itself, degrade the existing performance?

In order to answer the previously mentioned Research Questions, the following objectives were stated:

Objective 1: Classification of WSNs applications, as WSNs performance goals are heavily dependent on the application used;

Objective 2: Creation of a taxonomic tree that characterizes the requirements of WSNs with controlled performance;

Objective 3: Study and propose new WSNs performance metrics that allow for the evaluation of a selected group of requirements belonging to the taxonomic tree developed; These metrics should enable real-time monitoring of the network;

Objective 4: Collect the performance data from the WSN, having in mind the specific restrictions of these networks and minimizing the overhead.

Next figure (Fig. 1-1) represents a global view of WSNs with controlled performance, together with the main issues necessary for their implementation.

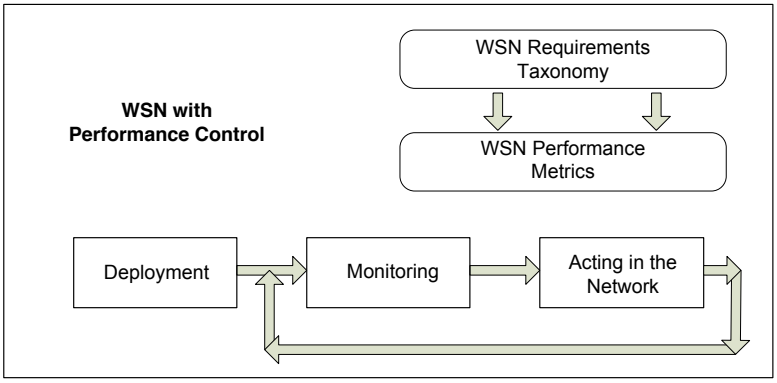


Fig. 1-1 – WSN with performance control.

A WSN with performance control uses specific performance metrics to assess its performance, which should be provided by a well-defined taxonomy that classifies all its performance requirements. After having metrics defined and indicative values for them, the network can be deployed. After deployment it is necessary to guarantee that the network continues to provide the necessary performance to its applications. To assess it the network must be monitored and eventually modified to continue to provide the performance needed.

The focus of this PhD thesis is the performance measurement in WSNs. The goal is to provide a common framework for measuring the *Quality of Sensing* (QoSensing) (see Section 2.4) of a WSN and to collect and debug that performance data without a significant overhead. The performance information collected should then be monitored and, if necessary, trigger subsequent corrective actions.

As a result of the work done, three main contributions resulted:

- **Taxonomy for the QoSensing requirements of a WSN.** A taxonomic tree of requirements that includes the specific needs of WSNs with controlled performance is proposed.
- **Metrics for the Network performance branch of the proposed taxonomy**. A generic set of metrics tailored to respond to the Network performance branch of the taxonomy created, is proposed. Different types of metrics are evaluated to assess which can deliver more efficiently the global QoSensing of the network. A small set to be used in generic industrial scenarios is also proposed.
- **Metrics collection protocol.** A new protocol that enables an efficient performance (and data) collection, using data fusion and considering the specific needs of performance monitoring in WSNs, is proposed. The protocol uses the specific WSNs characteristics to provide for a continuous evaluation of the global QoSensing of the network while providing mechanisms of alert when any malfunction is detected. The protocol also enables for simple debug operations.

1.4 STRUCTURE OF THE THESIS

The thesis is organized in six chapters and one appendix:

- Chapter 1, this chapter, presents the motivation for the undergone research, the thesis context, the initial research objectives, the contributions of the work done and finally the structure of the thesis.
- Chapter 2 presents some background information about WSNs, their application scenarios and the characteristics of QoS under WSNs. It also addresses some definitions used along the thesis. Finally, it gives a brief overview of the GINGENG project, for which this thesis contributed and that provided the grounds and knowledge necessary for some of the assumptions that led to the development of the proposed taxonomy and metrics.
- Chapter 3 proposes both a new classification of WSNs application scenarios and a new taxonomy for QoSensing requirements of a WSN. While applicable to any type of WSN, they address with more depth the case of WSNs with controlled performance. This is a contribution to the establishment of a new reference model that will serve as a basis to create and classify a new set of QoSensing metrics that characterize WSNs.
- Chapter 4 addresses WSNs QoSensing metrics. Different types of metrics are studied and their specific characteristics analysed in the context of their contribution to the evaluation of a WSN. A global set of metrics that fulfils the evaluation needs of the Network performance branch of the taxonomic tree presented in Chapter 3 is proposed. Also, a proposal of a framework that includes selected metrics to be used in generic industrial scenarios is also presented. The maintenance of the levels of

performance in a WSN, by using a continuous evaluation of some metrics and dynamically acting in the network, is also addressed.

- Chapter 5 proposes a metrics collection protocol, built to minimize the overhead implied in a continuous monitoring of WSNs with controlled performance. By using data fusion, collective metrics and alerts, it provides a low overhead while enabling the continuous assessment of the global QoSensing of the network and an immediate report of alerts.
- Chapter 6 concludes this thesis. It presents an overview of the work done and specifies its major contributions and results.
- Appendix A addresses details of the protocol proposed in Chapter 5 and presents some theoretical calculations about the additional number of performance packets generated.

Next, in Fig. 1-2, a graphical overview of the organization of the thesis, gives a deeper insight of how its objectives integrate with the phases of a WSN with performance control.

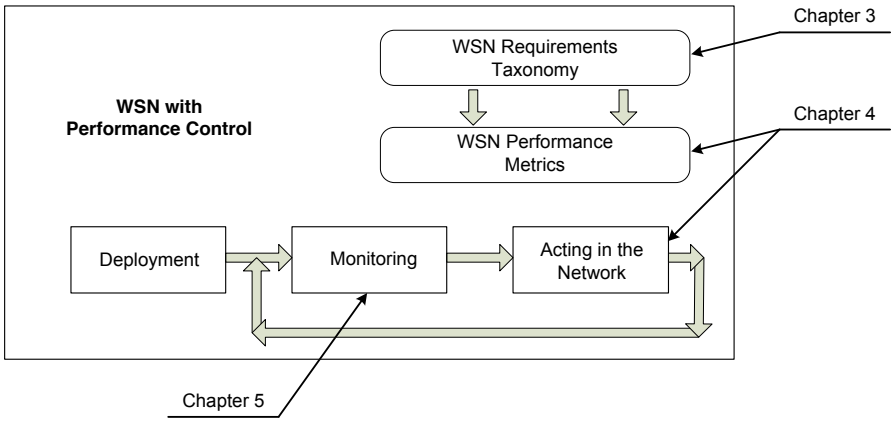
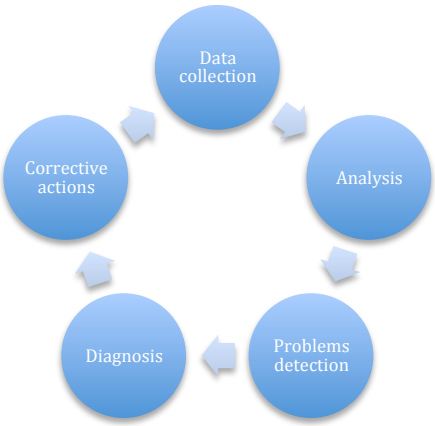


Fig. 1-2 – Thesis structure within the phases of a WSN with performance control.

Chapter

2

Performance in WSNs



In this Chapter, a general view of WSNs is provided, some research for QoS support in WSNs is introduced and some of WSNs open challenges stated. Also, some definitions used in the remaining text of the thesis, which relate with the specific case of WSNs with controlled performance, will be introduced.

2.1 INTRODUCTION

By performance it is meant not only the ability to accomplish a certain task but also how well it is done in comparison to pre-set standards of accuracy, completeness and speed. Performance can be measured by using specific metrics, with each metric evaluating a specific part of the process.

While the initial approach to WSNs did not target performance, the need to fulfil the demands of a new range of applications and scenarios like the ones existing in industrial plants and health monitoring, forced a new approach.

Early WSNs were focused in the random placement of sensors in an uncontrolled environment, relying on self-configuration and high levels of redundancy to achieve robustness, with no performance assurances. The new WSNs with controlled performance have different characteristics. These networks have specific performance targets that depend on the applications used and must support the existence of critical applications.

Many techniques have been proposed to enable QoS in traditional networks. However, to evaluate it, most of the parameters measured are the typical delay, delay variation, available bandwidth and packet loss, measured end-to-end. Additional metrics were defined to characterize the behaviour of the network, but rely on the same basic metrics [6].

WSNs, on the other hand, present a new range of applications and scenarios, which imply that a typical WSN must have to deal with new QoS demands and parameters. Also, the QoS requirements depend heavily on the application that is used. A WSN that targets precision agriculture will have in its QoS parameters the accuracy and precision of the measurements. An industrial plant will demand strict time restrictions, reliability and security as some of its QoS needs. A biodiversity mapping WSN will most certainly have coverage among its essential QoS necessities. Furthermore, WSNs present severe hardware restrictions such as processing power, energy, storage, communication capabilities, bandwidth, dynamic topology, non-uniform traffic, scalability or multiple sinks, raising the challenge of enabling these networks with QoS.

The new paradigm of QoS in WSNs was summarized in [7] using an holistic perspective. In this perspective, the QoS requirements in a WSN are viewed as a system that exceeds the sum of its parts, which results in a multiple impact that should be kept away from the users, being invisible to them, leading to the concept of “calm technology” [8] (Fig. 2-1).

Fig. 2-1 - CONET Consortium holistic perspective (adapted from [7])

Nevertheless, WSNs need some sort of management that is continuously working to guarantee that the network is always functional, providing for monitoring and maintenance. This is necessary for WSNs with or without controlled performance, although each with its own characteristics.

In order to control the performance of a WSN, a method to effective monitor and measure specific parameters is needed. The aim is to have feedback from the network so decisions can be made and corrective actions can be triggered. This monitoring must only include the necessary parameters, as collecting useless parameters results in extra traffic that must be treated and transmitted over the network, wasting valuable and scarce resources. The monitoring of WSNs must be used in three distinct phases - an initial deployment phase, a production phase (when the network is normal functioning) and a debug&recovery phase. In the first, monitoring is necessary in the acceptance tests, to guarantee that the network is working as planned, that performance targets are being achieved and that the protocols implemented are functioning properly. In the second, monitoring is used to evaluate the network performance targets in real-time. In this phase network QoS is measured to verify if the nodes are working as expected, all functionalities are evaluated and a network diagnosis is made. If necessary, a third phase of debug&recovery may exist, which triggers all the necessary debug and corrective actions.

2.2 WSNs

In this section a brief description of a WSN and its components is made.

2.2.1 Description and characteristics

A WSN consists of spatially distributed autonomous sensors used to monitor physical or environmental conditions, and to cooperatively pass their data through the network to a base station (Sink/Gateway). Additionally to sensors, these networks may also include actuators, i.e., devices capable of performing some kind of action in the surrounding environment, becoming Wireless Sensor and Actuator Networks. However, both types of networks are commonly addressed as WSNs. Data from the base station is then sent to users or applications for further processing or just for reporting (Fig. 2-2).

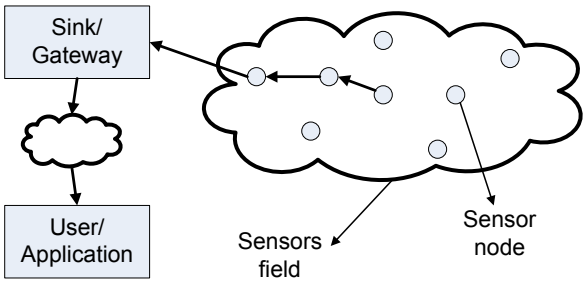


Fig. 2-2 –Wireless sensor network

Each of the sensor nodes is composed of the following components (Fig. 2-3) [9]:

- **Sensing unit:** converts physical phenomenon (e.g. heat, light, vibration) into electric signals by using a sensor and an *Analog-to-Digital Converter* (ADC);
- **Processing unit:** generally associated with a small storage, processes data and controls the functionality of all the components in the sensor node;
- **Transceiver unit:** connects the node to the network using wireless communications;
- **Power unit:** provides autonomous power for the node operation (e.g. batteries);
- **Location finding system (optional):** enables the node to find its global position or its position relative to its neighbours;
- **Mobilizer (optional):** moves the sensor node when it is required;
- **Power generator (optional):** enables the node to provide for its own energy (e.g. solar panels).

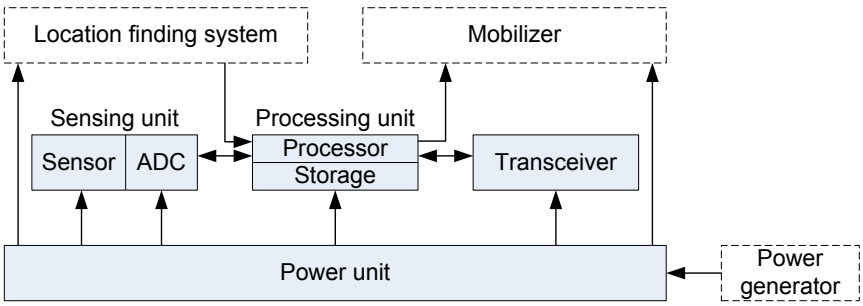


Fig. 2-3 – Sensor node components ([9])

Common sensor nodes are typically limited in power, computation capabilities and storage. These characteristics are even more aggravated by their low cost and small sizes. Also, as these devices run on batteries or use energy-harvesting schemes to obtain additional power, while operating with minimum or no external intervention, they rely on power saving schemes to prolong their lifetime, usually operating in low duty cycles. In consequence, all these characteristics normally translate in restrictions on the software that can be run, on limited communication possibilities (with limited ranges), and on poor localization features. In addition, the applications and scenarios used imply many times that sensor nodes are deployed in remote or hazardous places, subject to unpredictable environments that not only contribute for the number of hardware failures, but also difficult communications.

Sink nodes (multiple sinks may exist in a network) are usually more robust and with fewer limitations. They have improved computation power and memory size, sometimes being directly connected to power sources.

2.2.2 Scenarios and applications

Generic deployment scenarios can be divided in two groups. The first can be referred as a best-effort WSNs deployment (the typical scenario), and the second as WSNs with performance control deployment, whose characteristics are summarized in Table 2-1. Scenarios tailored to run specific applications may present characteristics from both of the main groups, being called hybrid deployment scenarios.

TABLE 2-1 – MAIN GROUPS OF WSNS DEPLOYMENT SCENARIOS

	Best-effort WSN scenario	**WSN with performance control scenario**
Environment	Uncontrolled	As controlled as possible
Deployment	Random placement of nodes	Studied placement of nodes
Topology control	Relies only in self-configuration	Has a defined topology that may have small changes along network lifetime or to support mobile nodes
Redundancy of nodes	High	Depends on the requirements of the specific applications used but is normally limited
Performance assurance	No	Yes
Traffic	All traffic has the same treatment	Different traffic may have differentiated treatment

While the initial applications of WSNs were designed to work in best-effort scenarios, that trend is changing. Attracted by the low deployment and maintenance costs, deeper control possibilities and high flexibility in the adaptation to changing business models, new application scenarios are emerging. However, for most of these new scenarios performance guarantees are essential. WSNs with performance control aim to respond to the new possibilities fulfilling the needs of real-world applications where performance is a critical issue.

Industrial scenarios are an example of the new demands from WSNs. In order to achieve high levels of performance, large industrial sites built extensive wired networks to connect all the control and sensing devices. While this solution works as expected in environments where everything was planned from the beginning, problems arise when new sensor or actuators must be added to control new processes or enhance existing ones. In this case, costs and difficulty of implementation are higher, and implementation time is longer, when compared to a WSN implementation. However, the WSN must assure the same reliability and performance control as the wired network, an area that has not been sufficiently focused by the research community.

Typical WSNs applications that run in typical WSNs scenarios include disaster relief operations, environmental applications (e.g. biodiversity mapping, forest fire detection), military operations (e.g. battlefield surveillance, reconnaissance of opposing forces, targeting, biological/nuclear/chemical attack detection). Common WSNs applications that run in WSNs with controlled performance scenarios include intelligent infrastructures (e.g. buildings, bridges), machine surveillance, environmental applications (e.g. precision agriculture), health-care (e.g. tracking of doctors and patients inside hospitals, drug administration, monitoring of physiological data), industrial applications, home and office (e.g. smart environments, home and office automation).

2.3 QoS IN WSNs

As applications for WSNs are expanding to new scenarios, the need of QoS is growing. The network not only has to transmit data but also has to satisfy the service requirements of each different application. However, distinct applications have different requirements. Some applications, like data gathering applications need reliable transmission but may not have any time constraints. In opposition, event driven applications normally require strict time constraints in order to respond in time to each event. Despite the fact that different applications may have different QoS demands, a common mechanism to measure the QoS being provided, to collect the performance data and to continuously monitor that performance, must exist.

2.3.1 Approach to QoS measurement in non WSNs

What is QoS?

Quality of service is a broad term used to describe the overall experience a user or application will receive over a network. *International Telecommunication Union's Telecommunication branch's* (ITU-T's) defines QoS as the "totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs" and also as "the collective effect of service performances, which will determine the degree of satisfaction of a user of service"[10]. A more practical definition is given in Wikipedia [11], which

defines QoS as "the ability to provide different priority to different applications, users or data flows, or to guarantee a certain level of performance to a data flow".

The QoS requirements of users connected to a data network are a statement of the level of quality the users expect to get from the applications they run or from the services they subscribe. This level of quality can be understood using two different approaches. In the first, QoS refers to the quality as perceived by the user/application, while the other refers to the measurable quality that the network offers to the application/user. An example of video streaming may be used to clear the difference. While the video transmission may be characterized by typical parameters like delay, jitter, bandwidth or packet loss, that are easily measurable, the user experience may be affected differently depending on the type of frames that are affected, expressing a different perceived quality. To avoid confusions when using the term in the context of WSNs, QoSensing (defined in 2.4) will be used along this thesis.

Measuring QoS

The need for QoS in traditional wired networks emerged from the necessity to guarantee a quality level higher than best-effort services could provide. At the same time, it was necessary to quantify or, at least, identify, if the service that the user was paying for was the one he was getting. The *Service Level Agreement* (SLA) made with the network operator had to be based on a mutual understanding of how the quality of that service was going to be measured.

Some of the most important QoS parameters that are measured and monitored are network availability, bandwidth, packet delay, jitter and packet loss. Network availability can have a tremendous impact over QoS, even during small periods of time, due to the discontinuity of the service. To prevent this from happening network operators need to have redundant equipment. As for bandwidth allocation, we must differentiate two different types: available bandwidth and guaranteed bandwidth. Available bandwidth means that the user can have a peak of bandwidth of certain width but that amount is not guaranteed in the SLA contracted with the network operator. This is the case of common *Asymmetric Digital Subscriber Line* (ADSL) or cable networks. The user may, in case of low traffic, get the maximum bandwidth, but there will be times when this bandwidth will not be achieved. SLA with Guaranteed bandwidth is usually more expensive since the network operator ensures that a minimum bandwidth width and a burst bandwidth are always available. Delay is the time datagrams take from the source to destination and depends not only on fixed delays (application delay, transmission over physical medium) but also from variable delays (queuing delays, contention with other traffic at each network node). Although small amounts of delay may be tolerated by applications, big delays may compromise applications such as voice and video. The measure of the delay variation between consecutive datagrams is called jitter. If jitter is not bounded, real-time and delay-sensitive applications (e.g. video streaming), where applications expect a fairly

constant rate between consecutive packets, can be greatly affected. Finally, packet loss, the number of packets lost in the network, via errors in the physical medium or by packet drop policies in network nodes due to congestion, as is easily understandable, can dramatically affect QoS.

2.3.2 QoS challenges in WSNs

QoS provisioning

Some of the features that challenge QoS provisioning in WSNs are resource constraints, platform heterogeneity, dynamic network topology and mixed traffic [12].

Resource constraints exist due to the inherent characteristics of typical WSNs. Sensor nodes are low-cost, with low battery, small memory size, limited processing and transmission capabilities. Although sensors hardware is in constant evolution, its characteristics are not expected to approach those of wired network nodes in a near future. All these constraints may cause network problems such unavailability of nodes, as well as a lack of processing power and/or communication resources in the network. Also, packets flowing in a changing topology, with battery fluctuations, are dropped due to small memories, and dispute the small available bandwidth. In consequence, delay grows and reliability decreases, leading to a poor QoS.

Platform heterogeneity derives from the different characteristics of available sensors and actuators. With different types of equipment, restrictions are not common and a minimum common denominator must be used.

Unlike wired networks, WSNs topologies are highly dynamic by nature, making this area one of the challenges to overcome. Nodes may move from one place to another, may be added, some die due to lack of battery. All this dynamics must be addressed by efficient protocols that must assure that QoS requirements are always guaranteed.

Mixed traffic exists because different applications produce different types of data, with different characteristics. Periodic and aperiodic data may coexist in the same network, with distinct QoS requirements.

QoS measurement

The measurement of the QoS in WSNs, is much more challenging and complex than in traditional wired networks [4], not only by the obvious hardware and software restrictions, but also because it is heavily dependent on the applications used, with different applications demanding different requirements. In fact, traditional QoS metrics like delay, jitter, bandwidth or individual packet loss are not enough to characterize and quantify the needed performance requirements of a WSN with controlled performance. Also, WSNs have specific constraints and requirements such as availability, connectivity, coverage, accuracy and security. Furthermore, as WSNs act as information systems,

metrics that relate both to communication and information issues are required (see proposed taxonomy in Section 3.4). In addition, WSNs benefit from having metrics built from measurements from different nodes, either to fully evaluate existing events [4] as to fully evaluate the performance of the global network.

Most of the existing QoS-aware protocols for WSNs rely on energy and connectivity issues, not dealing with many important issues like traffic differentiation, scalability, and critical data. Also, monitoring tools do not interact with the available protocols to permit a better use of resources, and the detection of error situations. Besides a QoS assurance, a QoS effective control is needed. To achieve control, a new set of metrics adapted to the WSNs specifics and created regarding real-time monitoring are also needed.

Next, some existing proposals that address QoS in WSNs are presented.

2.3.3 QoS proposals for WSNs

In spite of being a new research area in WSNs, many proposals were already presented to address the problem of QoS in WSNs. While some propose specific QoS mechanisms other achieve some sort of QoS indirectly. Next, some QoS based protocols for WSNs are presented [12] [13].

2.3.3.1 QoS-aware protocols

More than a protocol, Directed Diffusion [14] is a data dissemination paradigm for sensor networks. This approach tries to achieve QoS by eliminating data redundancy. On accomplishing the aggregation of data it saves energy, reduces the overall bandwidth and number of transmissions used, especially near the sink node. It works in data centric applications by using an attribute-value pair to describe the information sent by the nodes. It is based on a query-driven model, where the sink node broadcasts interests to the network, which are then sent from the nodes to the sink through multiple paths.

The *Sequential Assignment Routing* (SAR) [15] was one of the first QoS aware routing protocols developed for WSNs. This protocol builds multiple paths and chooses one according to energy resources and QoS, achieving fault tolerance at the same time. One of the drawbacks of this protocol is the amount of overhead necessary to keep the routing tables and state at each node, especially when a high number of nodes are considered.

Ngai *et al* propose a real-time communication framework [16], for event-driven applications in self-organized networks with sensors and actuators, which also enables reporting and actuator coordination. To accomplish for the minimization of network traffic and transmission delay, the event area is divided into smaller pieces of maps. The data is also aggregated and divided into different layers according to their importance.

QARP [3] is a QoS-aware routing protocol for WSNs whose objective is to meet QoS requirements for periodic, event-driven, and query based communications. The requirements include low latency for high priority packets, differentiated service packets, reliability, path repair in presence of failures, and energy saving.

2.3.3.2 Architectures for WSNs with controlled performance

It is not uncommon to build specific WSN deployments to work with specific applications. This happens due to the fact that each application has its own specific requirements, which are easier to satisfy with a dedicated deployment. However, in the case of networks with controlled performance, some approaches integrate the most common QoS requirements into an architecture that tries, by itself, to guarantee a predefined level of QoS.

To provide WSNs with performance assurances, several studies were made, resulting in new standards and WSNs architectures targeting wireless industrial communications, namely the ISA100.11a [17], WirelessHART [18] and GINSENG [5]. By defining specific algorithms and protocols, together with a specific architecture, they created WSNs that try to comply with industrial performance needs.

ISA 100.11a is a standard developed by the *International Society of Automation* (ISA) [19] under the description of "Wireless Systems for Industrial Automation: Process Control and Related Applications". It was designed to fulfil the industrial requirements of reliability and security, enabling monitoring, alerting, supervisory control and open and closed-loop control. It also addressed performance needs of applications such as critical monitoring and process control, with low latencies.

WirelessHART is an extension to the widely used wired HART communication protocol, maintaining compatibility with existing HART devices, commands and tools. The WirelessHART standard provides reliability (even in the presence of interferences), security, effective power management, and uses channel hopping and *Time Division Multiple Access* (TDMA) to ensure delay-controlled communications between devices on the network, complying with the needs of industrial environments.

GINSENG [5] – Performance Control in Wireless Sensor Networks, under which part of this thesis was made, developed a novel WSN with controlled performance for use in a range of industrial environments. This WSN meets application-specific performance targets, integrates with industry resource management systems, and was tested in a real industry setting, where performance was critical. To accomplish these objectives the project adopts a deterministic deployment of sensors together with new software and algorithms, including a TDMA *Medium Access Control* (MAC) layer to provide for strict delay bounds.

2.4 DEFINITIONS USED

To facilitate the understanding of the concepts used in these thesis, some definitions will now be presented.

WSNs with controlled performance and Best-effort WSNs

The term "Best-effort networks" is widely used to refer a network where QoS in not guaranteed and traffic does not have any differentiated treatment. Applying the concept to WSNs and considering performance demands, WSNs can be divided in two groups that will be hereafter referred to as WSNs with controlled performance and Best-effort WSNs. The first group is used in critical scenarios such as health applications, industrial and military facilities where there is a need for a certain level of performance. The second is used in non-demanding performance WSNs such as environmental surveillance and agriculture monitoring.

QoSensing

Two concepts are broadly used to express the service provided by a network and its applications: *Quality of Service* (QoS) and *Quality of Experience* (QoE). The first is used to describe the overall experience a user or application receives over a network, and is normally associated to a strict number of performance metrics used by traditional networks. The second provides an evaluation of human expectations and satisfaction with a specific service or application. To distinguish from both, the term *Quality of Sensing* (QoSensing) will be used along this thesis. QoSensing will be defined as the overall performance of a system, used for sensing or acting in the environment, which affects the quality of the information sensed and provided by the service. The measurement of the QoSensing is not dependent on subjective opinions from users but focuses on the quality of the information gathered and transmitted. It has a larger scope that QoS and is more objective and specific than QoE.

TABLE 2-2 – QOS, QOE AND QOSENSING CHARACTERISTICS

QoS	QoE	QoSensing
Evaluates the service by measuring several parameters of the network service, such as availability, error rates, loss, bandwidth, throughput, transmission delay and jitter.	Evaluates the service by using a subjective measurement of a customer's experiences with the service.	Evaluates the overall performance of a system, used for sensing or acting in the network.
Focus in the network parameters of the transmission.	Focus in the entire service experience.	Focus in network parameters and in the quality of the sensed information.
Objective measurements.	Subjective measurements.	Objective measurements.

2.5 THE GINSENG PROJECT

The participation in the GINSENG project provided a case study for this thesis and also valuable insights of a real deployment scenario, for which metrics were studied and proposed. In this section, an overview of the GINSENG project and its architecture are presented. Also, the work done under the project is detailed and its results specified.

2.5.1 Introduction

WSNs started to be applied in scenarios where reliability and performance control were not relevant. In these scenarios nodes were placed randomly and in high numbers, using self-configuration networks, recurring to high levels of redundancy and did not assure any QoS. However, to permit the adoption of WSNs in a broader set of scenarios, reliability and performance control are a primary demand. In fact, in most application domains nodes are deployed in a planned manner, close to places of interest and demand strict performance goals. One of these scenarios is the industrial environment. In this particular scenario, specific machinery, pipes, actuators, buildings that need to be monitored and controlled, coexist. Not only the placement of WSN nodes is carefully planned but also the end-user expectation in terms of reliability and latency in the response to events is very high, corresponding to the service levels that are provided by the wired technologies. The goal when adopting WSNs in industrial environments is to achieve both savings in deployment and maintenance costs and increased flexibility to adapt for changing businesses, while not lowering the service levels provided.

GINSENG – Performance Control in Wireless Sensor Networks [5], was a project funded by the European Union, under the *Seventh Framework Programme* (FP7), from September 2008 to February 2012. It had the participation of Universidade de Coimbra (Portugal), National University of Cork (Ireland), Petrogal SA (Portugal), SAP AG (Germany), Swedish Institute of Computer Science (Sweeden), Technische Universitaet Carolo-Wilhelmina zu Braunschweig (Germany), University of Cyprus (Cyprus) and Lancaster University (United Kingdom). GINSENG addressed the scenario of industrial environments where deployment is carefully planned, not entirely self-configuring, with low levels of redundancy and in which the performance is controlled and constantly monitored to ensure for strict application service bounds, especially for latency and reliability. As stated in the project Description of Work document, the "*goal is a wireless sensor network that will meet application-specific performance targets, that will integrate with industry resource management systems, and that will be proven in a real industry setting where performance is critical*" [20]. A general approach schema is depicted in Fig. 2-4.

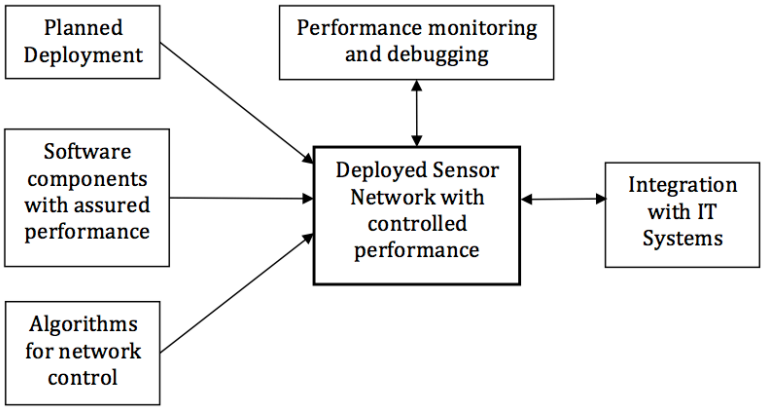


Fig. 2-4 - GINSENG approach ([20])

To fulfil the objectives, GINSENG required several innovations and a new approach to WSNs research. Firstly, "*GINSENG adopts a planned approach for sensor node deployment as a way to enable performance control*". Secondly, "*GINSENG relies in software components with assured performance, including operating systems and protocols for radio medium access*". Finally, "*GINSENG provides a set of algorithms that ensure control with respect to network topology and traffic*". These three components enable the possibility to deploy sensor networks with assured performance. In order to deal with uncertainties, GINSENG also provides mechanisms and tools to perform performance debugging of deployed systems. [20]

The development of the GINSENG project was divided in six different work packages: WP1 - Design and Algorithms for Performance-controlled Wireless Sensor Networks, WP2 - Network Elements and Debugging Tools for Performance-controlled Wireless Sensor Networks, WP3 - Middleware and system integration, WP4 - System Demonstration and Evaluation, WP5 - Dissemination and Exploitation, and WP6 - Project Management. Under this thesis the main contributions were given in WP1 and WP5, mainly under what concerned Task 1.2 – Measure of performance.

2.5.2 Scenarios

The test bed of the project, the Petrogal oil refinery at Sines, Portugal, is a complex industrial facility that needs careful monitoring and control of operations. About 35,000 sensors and actuators were used in the refinery to perform monitoring of industrial operations such as leakage detection, measurement of pressure in pipes, measurement of fluid levels and of the overall environment. In the oil refinery three subsystems exist for the monitoring and control of the plant: the *indicatory system*, the *semi-automatic control system*, and the *automatic control system*. The first is used to provide the control center with information about status and faults of equipment and generic aspects of the environment. Within this system, information flows one way from the in-field sensors to the control center. The second system controls different items of the refinery by including

actuators that receive commands sent by an operator as a response to data arrival from sensors. The third system is used to deploy automated control loops within the refinery. The system is similar to the previously described semi-automatic control system but commands to actuators are sent automatically upon receiving sensor data. Each of the systems described has different time and reliability boundaries. As all industrial plants have indicatory, semi-automatic control and automatic control systems, and similar requirements to those in the refinery, the three classifications of systems presented, should apply to any industrial plant. Therefore, it should be possible to apply the solutions found for these scenarios to the more general case.

Specifically, five scenarios were tested in the refinery. The Production Monitoring scenario (Fig. 2-5) is an indicatory system where sensors are deployed throughout the plant to monitor various aspects of production to help control center technicians on production decisions.

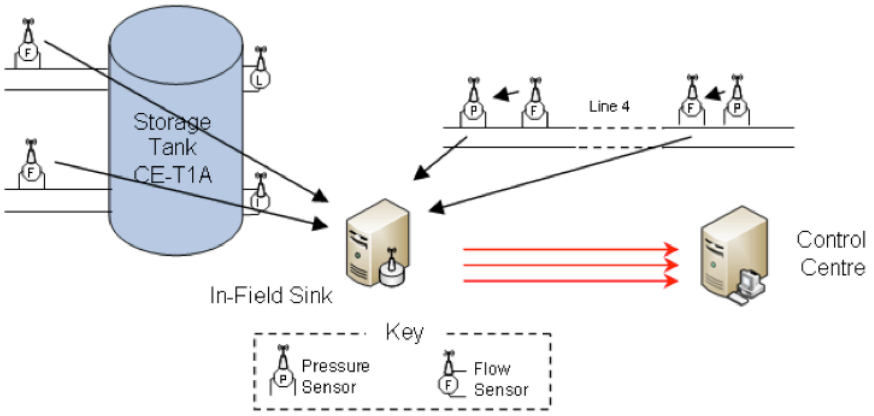


Fig. 2-5 - Production Monitoring scenario ([21])

In the Production Control scenario (Fig. 2-6), control center technicians make decisions and alter various aspects of production based on information that is received from the sensors. In addition to the sensors seen in the previous scenario, actuators are also included that can switch pumps, mixers or close valves. This scenario is an example of the semi-automatic control system.

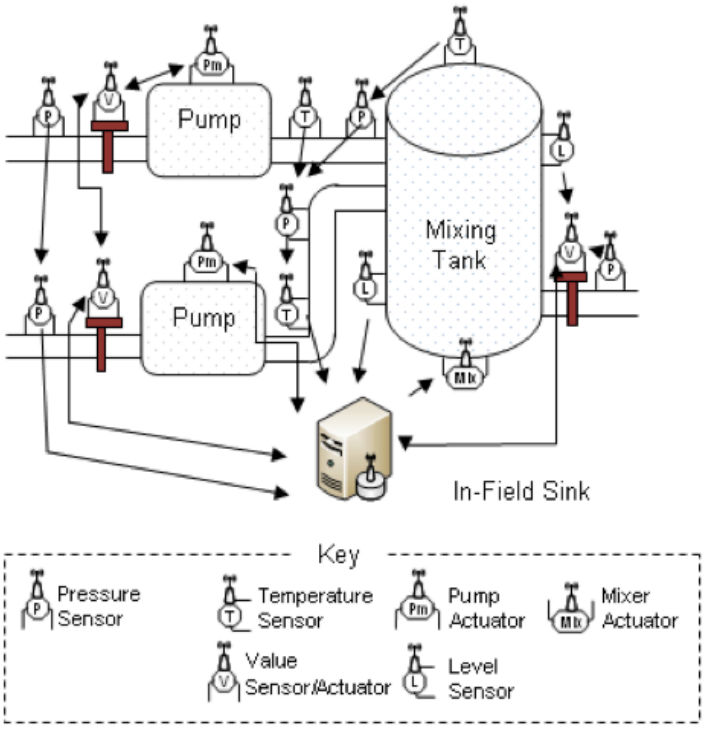


Fig. 2-6 - Production Control scenario ([21]**)**

The Production Monitoring and Control scenario (Fig. 2-7) is similar to the semi-automatic system seen in the previous section but includes automatic processes. Automatic processes are closed-loop systems where actions are performed automatically in the presence of certain conditions.

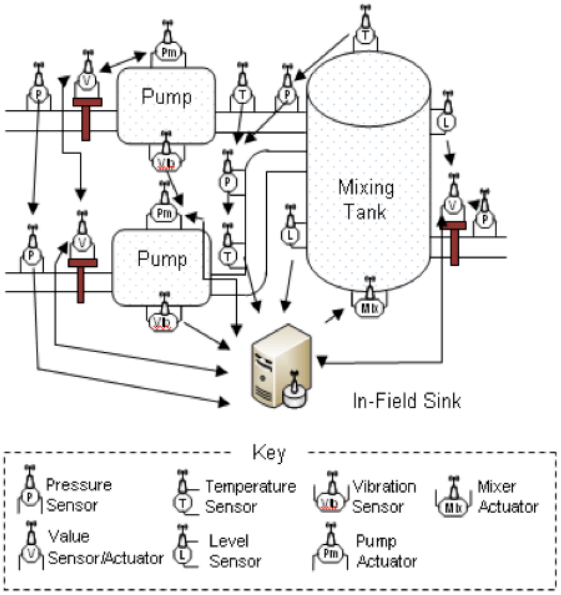


Fig. 2-7 - Production Monitoring and Control scenario ([21]**)**

In the Pipeline Leak Detection scenario (Fig. 2-8), oil pipelines are surveyed for leaks. Sensors and actuators will be deployed that monitor for leaks and then close valves to

reduce emissions. This scenario is an example of the automatic system and has a specialized linear network structure where nodes can be a large number of hops away from the sink.

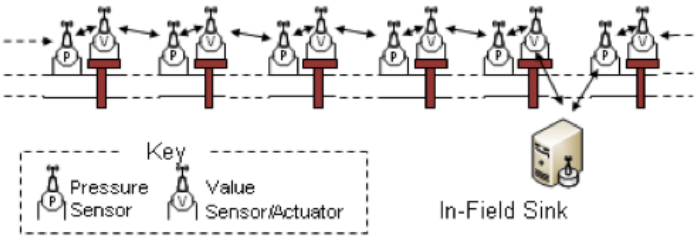


Fig. 2-8 - Application scenario (Pipeline leak detection) ([21]**)**

Finally, the Personnel Safety scenario (Fig. 2-9) monitors employees that enter in hazardous areas. By using orientation sensors attached to employees, their position can be monitored and alarms can be signalled when an employee is lying on the floor. This scenario is a specialized case of the indicatory system that includes mobile nodes.

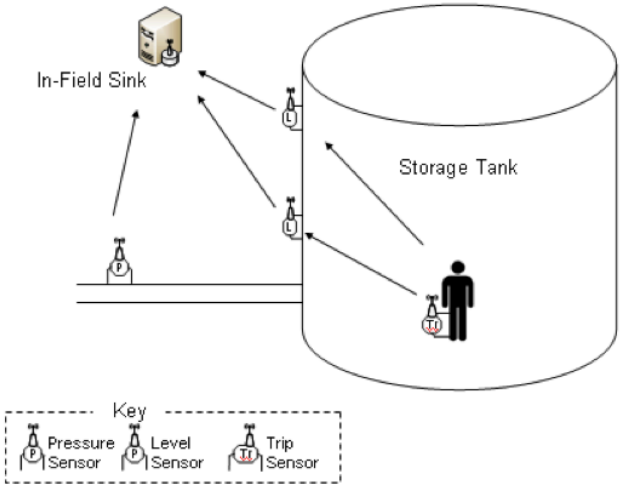


Fig. 2-9 - Personal Safety scenario ([21]**)**

All these scenarios provided for a rich source of information, and were used as reference in the development of this thesis.

2.5.3 Architecture

After analysing the requirements of the application scenarios (that can extend to other environments besides the industrial), a specification of a functional architecture that provided for performance control in wireless networks was defined (Fig. 2-10).

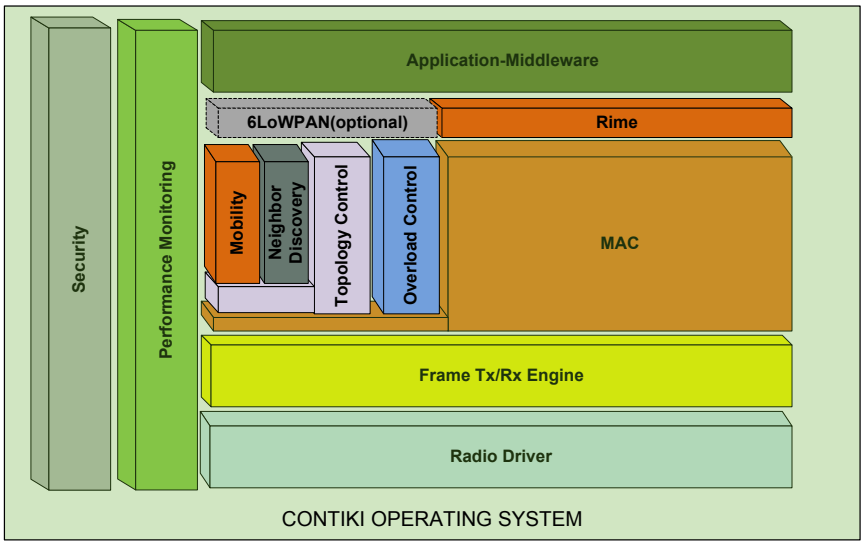


Fig. 2-10 - GINSENG functional architecture ([21])

The functional architecture of GINSENG includes several layers. The Physical layer (*Radio* and *Frame Tx/Rx Engine*) is responsible for the frequency selection, modulation and data encryption. The CC2420 radio was used for testing purposes. The frame transmission and receiving engine makes all the necessary operations for each TDMA slot. The TDMA-based MAC Layer (GinMAC) provides the access of the nodes to the shared medium ensuring that all performance requirements defined by the applications are satisfied. This protocol was specifically developed for GINSENG and uses a pre-dimensioned virtual tree topology and hierarchal addresses. It interacts closely with *Overload Control* and *Topology Control* layers. *Overload Control* deals with issues of traffic congestion. *Topology control* aims to improve energy efficiency and network performance and manages the network tree topology, including the joining and leaving of nodes. It also provides for slot allocation, transmission power decisions, tree optimizations and maintenance of the tree. *Topology control* also includes *Mobility* and *Neighbour discovery* modules. *Neighbour discovery* allows for the advertisement of nodes to their neighbours while *Mobility* exists to guarantee reliability in scenarios like Personal Safety Monitoring. The *IPv6 over LoW Power Wireless Personal Area Networks* (6LoWPAN) [22] adaptation layer adapts *Internet Protocol* (IP) version 6 traffic headers to make more free space available in the payload of 802.15.4 frames. It was found to be optional due to memory restrictions. In the case of its absence a light communication stack provided by the Contiki Operating System [23] was used – Rime [24]. The *Application-Middleware* layer hides all the hardware and software of the lower layers to the end-user, providing an easy interface to work with. *Security* addresses issues of cryptography and key management to establish protection mechanism in the WSN. *Performance monitoring* is essential to guarantee that the initially specified performance is being maintained and, in case it is not, debugging can be executed.

2.5.4 WP1-Task 1.2-"Measure of Performance"

The main contribution of this thesis work to the project GINSENG was made under Task 1.2 – Measure of Performance. The objective of Task 1.2, whose task leader was *Faculty of Sciences and Technology of the University of Coimbra* (FCTUC), was to define and measure the performance of the proposed WSN. The task was divided in two sub-tasks, the first being the definition of the quality of service parameters relating to performance, and the second their effective measurement.

A summary of the interaction of Task1.2 and other GINSENG tasks is presented next.

- Task 1.1 (Sensor Network Architectures and Paradigms): provided the identification of topologies and scenarios where the performance was to be measured.
- Task 1.4 (Overload Control): supplied metrics to help the overload control of the sensor network.
- Task 2.1 (Sensor Node Operating System): provided the interface for the monitoring applications with lower end layers.
- Task 2.3 (Sensor Node Lifetime Prediction): energy metrics provided a more accurate lifetime prediction.
- Task 2.4 (Performance Debugging): provided the metrics to detect the decay in performance and the need for debugging.
- Task 4.3 (Software Integration - Application level software): the control of metrics was part of the monitoring software integrated in this task.
- Task 4.4 (Tests and evaluation): provided valuable input and tested the measurement of the metrics developed.

As part of the solution, Task 1.2 also participated in the global evaluation and dissemination task.

- Task 4.4 (Tests & Evaluation): this task evaluated the overall performance of the solution developed and also Task 1.2 results
- WP5, as a whole, used the results of all other tasks and therefore also the results of Task 1.2.

The work done by the author of this thesis consisted in two parts. The first was to analyse the specific requirements of a WSN with controlled performance in the context of the project. As these requirements were application dependent, a taxonomic approach based on different application types including the ones chosen for GINSENG was made, and the same was done for the requirements. The aim was to categorize the needs of the WSN applications in order to understand their specific performance requirements, which would

then lead to the definition of the necessary metrics. The network parameters to be used were also investigated. Then, the next step was the definition of the metrics that were most relevant in the context of GINSENG specific scenarios. These metrics were discussed within the project and a small set with priority metrics was chosen. At the same time, a survey of monitoring tools was also made. The monitoring tools are essential for the measurement of the network parameters that will make the metrics. The most representative monitoring tools were analysed and compared.

The results of this work were released under Project Deliverables [25] and [21]. Parts of those results are also presented in Sections 3.6 and 4.9 of this thesis.

2.6 CHAPTER SUMMARY

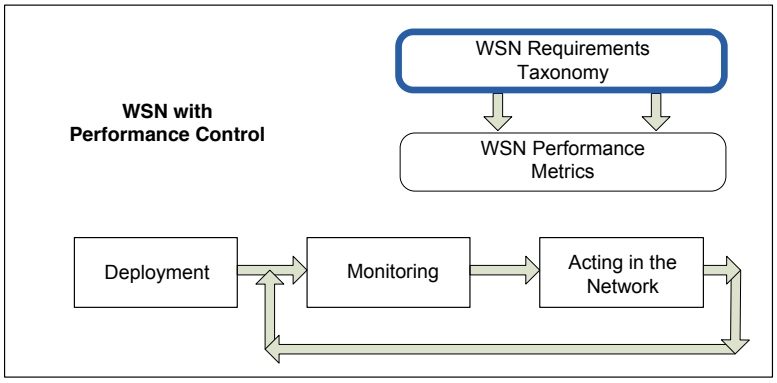
In this chapter, a general overview of WSNs characteristics, scenarios and applications was given. Also, the concept of QoS was introduced together with the specific challenges of QoS in WSNs and some existing proposals already made to address the problem. Some concepts used along the thesis, namely QoSensing and WSNs with controlled performance were also stated. Finally, it presented an overview of project GINSENG, for which this thesis contributed, especially under Task 1.2 (Measure of performance). Project GINSENG is used along this thesis as a reference and a case study to some of the solutions proposed.

Next chapter will address the characterization of WSNs and the proposal of a new taxonomy of requirements for WSNs with controlled performance.

Chapter

3

Taxonomic Proposal



The emergence of a new range of applications and scenarios that demand WSNs with QoSensing assurances and support, poses new challenges, the first one being the need to know which are the specific requirements of these new networks. However, QoSensing in WSNs, and a reference model to characterize it, are yet challenges to be met.

In this chapter, both a new classification of WSNs application scenarios and a new taxonomy for QoSensing requirements of a WSN are proposed. First, the analysis is focused in WSNs applications characterization. This step was necessary in order to fully understand all the characteristics and necessities that result from the fact that different applications and scenarios lead to different QoSensing requirements and restrictions. After, a taxonomic tree of WSNs QoSensing requirements is proposed. This is a contribution to the establishment of a new reference model that will also aid in the development of a set of QoSensing metrics that characterize WSNs (including the ones with controlled performance). Finally, the proposed classification and taxonomy are applied to selected projects.

3.1 INTRODUCTION

Taxonomy is the science of classification. Using taxonomy, specific taxonomic schemes (commonly addressed as taxonomies) classifying a reality or a concept are produced. The word taxonomy comes from the Greek *taxis* (arrangement) and *nomos* (law) [26]. Initially only used by Biology to classify living organisms, it is now used in many areas. Taxonomy uses taxonomic units addressed as taxa (groups) and taxon (singular items).

A taxonomic tree of the QoSensing requirements of a WSNs with controlled performance will help in the classification and in the analysis of these networks, providing the grounds to a definition and classification of WSNs performance characteristics. Furthermore, it will enable the future development of a new class of metrics, adapted to the specific needs of WSNs, which measure and control that performance. Finally, we expect this work to be part in the effort to standardize WSNs. In fact, part of this research results from the contributions of the author to the *International Organization for Standardization* (ISO), as expert, specifically in the ISO/IEC JTC 1 study group on Sensor Networks (International Organization for Standardization / International Electrotechnical Commission - JTC 1 SGSN Study Group on Sensor Networks - Oslo meeting, June 2009 [27]) that intends to study Sensors Networks, including the "*survey of generic Sensor network technologies and functions, general requirements, reference architecture and models, applications/services examples, and other Sensor Networks related areas*"[27].

Next section presents the related work on classification of WSNs, in its multiple perspectives.

3.2 RELATED WORK

A vast effort to list and classify WSNs has been made, with different authors adopting different perspectives when addressing the problem. A chronological view of these efforts is presented next.

In [9] Akyildiz et al. focus the design of WSNs and identify 8 constraints factors that they found essential: *fault tolerance*, *scalability*, *cost*, *hardware constraints*, *topology change*, *environment*, *transmission media* and *power consumption*.

Tilak et al. [28] address WSNs focusing on the aspects that influence the network protocol, fundamental part of their sensor network organization (that is viewed as consisting of an infrastructure, a network protocol and application/observers). The classification of WSNs is made according to different communication models (application and infrastructure), data delivery models (continuous, event-driven, observer-initiated and hybrid) and network dynamics (static or dynamic sensor networks - dynamic are divided in mobile observer, mobile sensors and mobile phenomena). They also propose some network performance metrics to evaluate WSNs protocols (energy efficiency/system lifetime, latency, accuracy, fault-tolerance and scalability).

In [4] Chen and Varshney address QoS support in WSNs, focusing on the QoS requirements imposed by the main WSN applications to the network. In the analysis, two different perspectives are used: *Application-specific QoS* and *Network QoS*. The first, Application-specific QoS, identifies the requirements of each application, such as coverage, exposure and optimal number of sensors that affect the deployment of the sensor network. The second, Network QoS, focus on the delivery of data through the network, as done by different types of applications, grouped by data delivery model (event-driven, query-driven and continuous). Specifically, end-to-end performance needs, interactivity, delay tolerance and criticality are analysed. As a response to the specificities of WSN applications requirements, they introduce new non-end-to-end QoS parameters, named as collective QoS parameters. Namely, they define collective latency, collective packet loss, collective bandwidth and information throughput.

In [29] Hill et al. present a functional taxonomy for WSNs. They propose to divide WSNs platforms by device class in a tiered architecture that outlines the main characteristics of the four classes of nodes - Special-purpose sensor nodes, Generic sensor nodes, High-bandwidth sensor nodes and Gateway nodes.

Another functional approach is made in [30] by Cheekiralla et al., classifying wireless communication devices (including a broad range of devices that span from RFID devices to cell phones) based on their functionality, being WSN devices a part of the group.

In [31] Römer and Mattern propose a taxonomy of twelve dimensions to define the WSN design space. The proposed classification allowed for the characterization of WSN applications based on the technical properties of the underlying network. The dimensions proposed were: “Deployment” (random or manual; one-time or iterative), “Mobility” (immobile, or partly, or all; occasional or continuous; active or passive), “Cost, Size, Resources and Energy” (brick or matchbox or grain or dust), “Heterogeneity” (homogeneous or heterogeneous), “Communication Modality” (radio, or light, or inductive, or capacitive, or sound), “Infrastructure” (infrastructure or ad hoc), “Network Topology” (single-hop, or star, or networked stars, or tree, or graph), “Coverage” (sparse, or dense, or redundant), “Connectivity” (connected, or intermittent, or sporadic), “Network Size”, “Lifetime” and “Other Requirements” (e.g. real-time constraints, robustness). This version was later refined and simplified in [32] by Rocha and Gonçalves that also included a dimension of “Application specific needs”. The proposed dimensions were “Application requirements and Environment interaction”, “Network Dynamics”, “Cost, Size, Resources, Energy and Lifetime”, “Heterogeneity and Complexity”, “Infrastructure and Communication Modality”, “Network Topology and connectivity” and “Quality of Service”.

In [33] Iyer et al. provide a classification based on WSNs application objectives, traffic characteristics and data delivery requirements. They select as dimensions for the WSNs

applications taxonomy “Event detection and reporting”, “Data gathering and Periodic Reporting”, “Sink-initiated querying” and “Tracking-based applications”.

Also, using an application perspective, Ruairí et al. [34] identify nine dimensions to differentiate application classes and provide a requirements based taxonomy for WSN applications. The dimensions proposed were: “Spatial-Resolution”, “Temporal-Resolution”, “Lifetime”, “Latency” (negligible, or moderate, or strict), “Coverage” (partial, or full, or redundant), “Sensed Phenomena” (single discrete-target, or multiple discrete–targets, or single distributed, or phenomena, or multiple distributed phenomena), “Bandwidth” (episodic-small, or episodic-large, or continuous-small, or continuous-large), “Control” (external, or central, or distributed) and “Users” (single, or competitive, or cooperative, or collaborative). This classification focuses mostly on applications properties not including technical properties of the underlying network.

In [35] Bai et al. propose a taxonomy of WSN applications relating to the way their characteristics affect the complexity of the programming language used. The dimensions proposed were: “Mobility”, “Initiation of sampling”, “Initiation of data transmission”, “Actuation”, “Interactivity”, “Data interpretation”, “Data aggregation” and “Homogeneity”.

In [36] this thesis author proposes a novel taxonomy of requirements for WSNs with QoS. Also, a new proposal for classification of WSNs applications scenarios was made. The WSN requirements taxonomy aggregates previous studies and classification of WSNs, and includes a new focus on QoS requirements. Each of the requirements should be mapped into usable metrics that define completely (or as completely as possible) the QoS provided by the WSNs. The taxonomy presented was the initial version of the taxonomy described and proposed later in this chapter.

The development of taxonomies for WSNs continued with focus in programming languages and design.

In [37] Mottola and Picco identify the most common differences in WSNs that affect the design of programming approaches. They propose five dimensions to classify WSNs applications: “Goal” (sense-only or sense-and-react), “Interaction Pattern” (one-to-many, or many-to-many, or many-to-one), “Mobility” (static, or mobile nodes, or mobile sinks), “Space” (global or regional) and “Time” (periodic or event-triggered). Based on this classification, Oppermann et al. proposed yet another classification in [38] to try to fully capture all the aspects relevant in WSNs design. The 11 dimensions proposed were: “Goal” (sense-only or sense-and-react), “Sampling” (periodic or event-triggered), “Sensed phenomenon” (single or multiple; discrete or distributed), “Data rate” (low or high), “Heterogeneity” (sensors or architecture), “Mobility” (mobile nodes or mobile base-station), “Connectivity” (connected, or intermittent, or sporadic), “Processing” (operations of filtering, or compression, or aggregation, or tracking, or event detection, or

classification, or decision making that can be a made in a node, or network, or gateway, or server), “Storage” (caching or persistent occurring in node, or network, or gateway, or server), “Services” (localization, or time synchronization, or authentication, or encryption, or reprogramming, or reconfiguration) and “Communication primitives” (single-hop unicast, or multi-hop unicast, or single-hop broadcast, or flooding, or collection, or cluster).

Despite all the work already done to characterize WSNs, most of them present limited scopes and application areas, and at the time of our studies and proposals did not fulfil the requirements of a general classification that includes the demands of WSNs with controlled performance. The pretended taxonomic tree of requirements should not only serve as a classification of the general requirements that the underlying network should provide applications, but also as a basis for the future development of specific metrics to measure each of the found requirements.

3.3 CLASSIFICATION OF WSNS APPLICATION SCENARIOS

Applications have a central role in the definition of what to expect from a WSN, with different types of applications demanding different WSNs requirements. Therefore, to build a taxonomy of WSN QoSensing requirements, it is essential to start by analysing and classifying all possible WSN application scenarios. Besides theoretical analysis of the problem, a thorough analysis of the GINSENG test bed real scenarios and corresponding applications was made (see Section 2.5.2), as they are representative of an advanced industrial environment with multiple needs. While some classifications already exist (e.g. [28][4]), they do not address the new application scenarios created by the new WSNs with controlled performance that can operate in critical environments. Namely, this new WSNs that are emerging, are constituted by static and mobile nodes, with random or deterministic deployments, in networks that not only serve the purpose of sensing the surrounding environment but that may also act on it within bounded time constraints. Maintained in these new networks are the data delivery models that are shared with old WSNs classifications. The resulting classification proposed, that aggregates and upgrades previous ones, is depicted in Fig. 3-1.

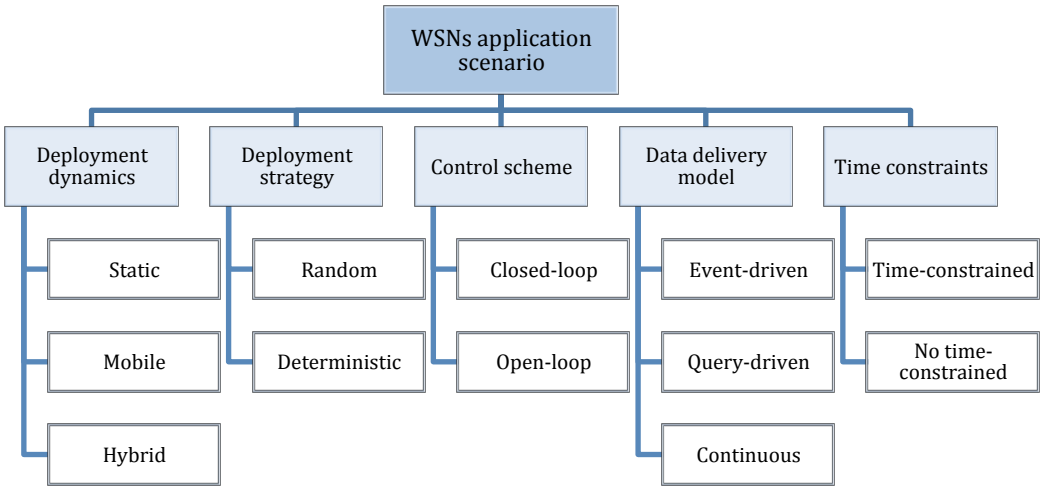


Fig. 3-1 – Classification of WSNs application scenarios.

The classification considers five different dimensions to classify WSNs applications: *deployment dynamics*, *deployment strategy*, *control scheme*, *data delivery model* and *time constraints*. Each dimension can be classified according to its lower level options.

The first dimension concerns the deployment dynamics. WSNs applications may be deployed using only static nodes, only mobile nodes or by using a mixed scenario that uses both static and mobile nodes. Examples of a static deployment can be a seismic monitoring application, and a military target tracking application could benefit from a mobile deployment.

The second dimension consists of the deployment strategy that may be random or deterministic. A random deployment is a deployment where there is no controlled placement of the nodes. Example of this scenario is a group of sensors thrown by a plane in a field. A deterministic placement of nodes occurs when the location of each node is previously planned. Examples of a deterministic placement occur in typical industrial plants where nodes are carefully located next to their sensing/acting targets. Mobile nodes can also have a deterministic or random deployment depending on their initial placement and on the way they move. While a hybrid model consisting of nodes randomly placed and nodes whose location is previously studied may in theory exist, it is very uncommon. In these specific cases they normally constitute different WSN entities. Therefore, a hybrid scenario was not included in the classification tree.

The third dimension is the control scheme. This dimension is divided in open-loop applications, when there is only one way for the data transfers between each sensor and the sink or gateway node, or closed-loop applications, when there is communication in both ways (Fig. 3-2). An open-loop application does not involve feedback, while a closed-loop application may use feedback to enable a response from the system. An example of an open-loop application is the monitoring of temperatures from different

sensors. A closed-loop application could involve a sensor whose data, after processed by the sink, could determine an action by an actuator.

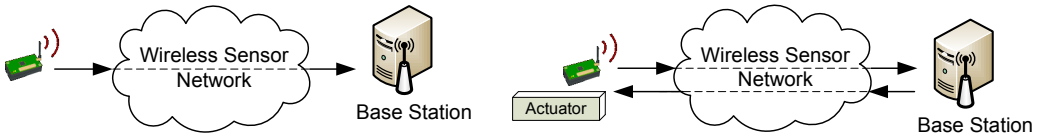


Fig. 3-2 – Open-loop (left) versus Closed-loop (right) control schemes.

The fourth dimension corresponds to the data delivery model. This dimension is classified in event-driven, query-driven and continuous delivery models. Eventually, a hybrid model combining two or three of the referred models may also exist, as different models may be used for different data being delivered in the same network by the same application. The first model, event-driven, is used when a predefined event triggers the sending of data by a sensor node. Examples are high temperature alerts and gas leaks detection by nodes. The second model is the query-driven, which represents applications in which data is pulled on-demand. In this model the nodes only respond to queries made from other nodes or by the sink. Examples of this data delivery model are video surveillance WSNs, on-demand nodes positioning retrieval and all the non-periodic data retrieval in response to specific requests. The last model, continuous model, represents applications that require that sensor nodes periodically report their data do the sink node, normally referred as data-gathering applications. The period can approach real-time data reporting (e.g. voice or image streams from sensors, continuous measurements of a real parameter such as temperature) or just a timely reporting of data (e.g. agriculture and wildlife scenarios).

Finally, time constraints dimension classifies the application as being time-constrained, or no time-constrained. A time-constrained application is an application that demands strict time boundaries to receive information from the nodes and to eventually send feedback to sensors or actuators. An application where response time is not crucial may be considered as no time-constrained. Examples of time-constrained applications are industrial critical, health monitoring and real-time applications.

An example of the application of the proposed classification to GINSENG and other application scenarios is shown in Section 3.5.

3.4 TAXONOMY OF THE QOSENSING REQUIREMENTS OF A WSN

Several classifications exist for WSNs (see Section 3.2). However they focus in particular areas with restricted scopes and lack to include the needs of networks with controlled performance. The motivation to design a new taxonomy was the need to understand, using a user/application perspective, which requirements these networks have. This knowledge is of crucial importance to create the metrics to measure and evaluate WSNs.

As the performance requirements of a WSN surpass the notion of traditional QoS parameters, the term QoSensing will be used (see Section 2.4).

First, some general definitions of node, network, sensor, sink, area, user, phenomenon and event, in a WSN perspective, will be provided, as they are needed to understand the taxonomy to be developed.

- **WSN node –** Autonomous device with communication and processing capabilities, running on batteries and that can have sensing or actuating capabilities.
- **WSN –** Set of interconnected nodes sharing a common identifier, a network *id* or a network address prefix, and that are able to communicate with each other. Nodes can perform intra-mobility when they are moving inside the same area and keep the same identifiers, or inter-mobility when they move between different areas getting new global identifiers along the path. A network has one or more special nodes called sinks to where common nodes deliver data.
- **Sensor –** A sensor is a device that observes one or more phenomena, measuring the physical properties and the quantity of the observation, and converting the measurement into a signal. Is connected to the WSN node.
- **Sink/Gateway** – Special node for which the information is driven; it may be the last stop for the information collected from the network or it can be a gateway between networks.
- **Area –** The scope of a specific network. In each network the area or covered surface is dependent on the number of nodes, on the surrounding environment and on the respective radio strengths.
- **User/Application** – Final destination of the information gathered and processed by the network.
- **Phenomenon** – The entity of interest of the user, which is the focus of the sensing (e.g. a physical occurrence such as heat, light, motion, vibration, and sound). Multiple phenomena may be under observation concurrently in the same network.
- **Event** – A specific occurrence of a phenomenon that triggers an action from the node/network.

3.4.1 Taxonomy groups

On a first approach to the design of a taxonomic tree of requirements for WSNs with QoSensing, some groups of requirements were found critical: *Network dynamics*, *Security and Privacy*, *Deployment and Coverage*, and *General communications performance* (Fig. 3-3). However, WSNs are networks that not only transmit information between devices, but part of their performance comes from how they deal with the information that they sense and provide to the final user. In fact, sensor networks act as information systems that acquire, process and deliver information, not only as communication systems for information exchange. In consequence, a top-level division between a Network group and

Information group was made. Three additional groups to deal with information were also added: *Information gain*, *Information efficiency* and *Applications specific*. The resulting classification is depicted in Fig. 3-3. These groups will now be analysed in detail.

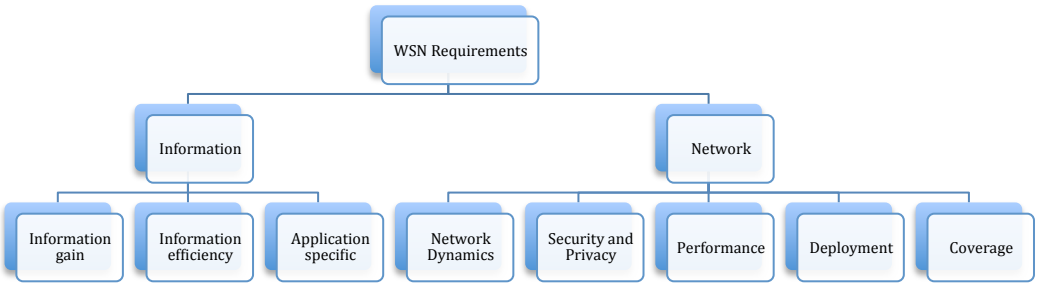


Fig. 3-3 - Approach to the taxonomy of a WSN with QoSensing

3.4.1.1 Information group

This group contains all the requirements that focus the network as an information provider. All the needs that relate to information quality, amount or efficiency, are part of this group.

3.4.1.2 Network group

This group includes all the requirements that relate to the communication service provided by the network, including its coverage of the environment to sense.

3.4.1.3 Information gain

The Information gain group includes all the requirements that address the amount and the quality of the information obtained from the data sensed. As examples, WSNs applications demand different cadences of data feeds from sensors, depending on the criticality of the information, require different precision in the data transmitted, demand different quality in the information (e.g. number of false positives and false negatives, false alarms [39]). The properties of the information produced in the nodes are essential to the evaluation of the QoSensing of the WSN.

3.4.1.4 Information efficiency

To the WSNs applications, besides quality of data, it is also an important requirement to assess the cost that the information has to the network. Information efficiency group includes all the requirements that impact the efficiency of the information collection regarding the resources used. An application should be able to specify its requirements in what regards the efficiency to be achieved in terms of resources (such as node activity time, node processing time or energy spent) while obtaining each piece of information.

3.4.1.5 Application specific

Each application has its own set of specific requirements, which are linked to its specific goals. This group contains all the requirements regarding the producing of information that are not considered essential to all networks and that do not fit either in Information gain or Information efficiency. The exact specific requirements inside this group are not specified, but a small set is used as an example.

Local storage is very important for applications where data is sent sporadically, either because the connections to other nodes are not always present or because it does not compensate to have a permanent flow of data [40].

Accuracy, the discrepancy between the real world values and the provided results, or the correct/optimal decision and the taken one within a given time interval, is a characteristic that affects applications that measure environment values. This requirement is mainly affected by the quality of the hardware used. Precision, the probability with which a given accuracy is achieved may also be important to applications that tolerate a specific amount of tolerance in the sensing done.

The localization resolution of nodes/phenomenon/events is other important requirement, used in applications such as target tracking [39].

For applications where in-network processing is essential processing power of the nodes is a requirement to consider. This requirement affects the hardware chosen for the network.

Applications that use networks where redundancy exists can specify a requirement such as the operational tolerance. This requirement sets the minimum conditions for operational use of the network (such as the minimum number of active sensors that the network must have to provide the applications with the necessary data).

3.4.1.6 Network Dynamics

Network dynamics includes all the physical movements that occur in the network and in its sensing target that affect the network QoSensing, and can be translated in the mobility of its components and in the connectivity of the network. Specifically, mobility in WSNs can be split into four main different fields [41] [27]:

- Mobility of the Sink/Gateway;
- Mobility of the Node;
- Mobility of the User;
- Mobility of the Phenomenon.

The main purpose of having mobility of the sink is to avoid the high cost of a long multi-hop network that implies the existence of nodes that only forward information. Mobile

sinks allow for the collection of information from sparse nodes, contribute for the overall energy saving, increase the coverage and reduce maintenance costs.

The mobility of nodes may be intentional as a node tracks with its own means a specific phenomenon, can be caused by external natural factors (e.g. wind, water) or by the fact that the node is itself attached to a moving host (e.g. nodes in the back of animals, nodes in vehicles, body sensors).

Mobility of users is related to the moving of the final recipient of the information gathered by the network. The user may communicate with infrastructure networks through the sink node or it can access each node directly if a non-infrastructure network is considered.

The mobility of the phenomenon is related with the displacement of the target of sensing. It implies the use of special applications such as in target tracking.

Independently of the mobility type, it increases the network coverage [42], lifetime and connectivity [43], being presented in several energy-aware WSNs algorithms and protocols.

Besides mobility, the connectivity of the network is also essential when assessing the dynamics of the network as it specifies if two different hosts can reach each other through the network.

3.4.1.7 Security and privacy

As WSNs may deal with sensible data, security and privacy must also be taken into account. Not only the data must be kept away from unauthorized user access but must also be protected from external attacks. Due to their nature, WSNs are vulnerable to several types of security attacks: eavesdropping (unauthorized overhearing of the WSN traffic), insertion (insertion of forged packets into the WSN), masquerading (identity forging by an unauthorized external entity), authorization violation (use of a service by an unauthorized external entity), loss of information (deletion of information in nodes or in transit), modification of information (change of the information in the nodes memory or while in transit) and repudiation (entities may be able to refuse responsibility for their actions) [27][44]. The taxonomy should consider a set of generic security services [27][45] that may be required by WSNs: *confidentiality*, *data integrity*, *availability*, *authentication*, *access control*, *accountability* and *authenticity*.

Confidentiality, integrity and availability, also known by the CIA triad or CIA triangle ([46]), are the core of information security. Confidentiality is designed to prevent sensitive information from being accessed by the wrong people and, at the same time, guarantee that it can be accessed by the authorized ones. It is important to guarantee confidentiality while data is being generated, transferred or stored. The most usual

method of ensuring confidentiality is data encryption, which should be implemented at different levels in the sensor network. Data integrity involves the guarantee that data is an accurate and unchanged representation of the original information. It is achieved by adding message integrity codes to the data. The most usual method of guaranteeing data integrity is through hashing. Availability is the capability to guarantee, with a previously defined degree of confidence, the normal operation of the all network. This guarantee should exist even in the presence of attacks or failures. It is usually achieved by having fault tolerant systems, redundancies and backups.

The next three security properties, authentication, authorization (access control), and accounting (AAA, pronounced "triple-A") pretend to answer respectively the questions "Who or what are you?", "What are you allowed to do?" and "What did you do?" [47]. Specifically, authentication is related to the capability of verifying or confirming the identity of a user or machine that operates in the network. Authorization involves the capacity to restrict the access of entities (users or machines) to resources (physical or computational). Finally, accountability involves the possibility of tracing all actions done to the data so that responsibilities can be demanded and users can rely on the information provided. It is normally achieved by company/organization internal and legal regulations.

Authenticity is the guarantee that the identity of a subject or resource is the identity claimed. Also, it should be possible to attribute it to its owner (non-repudiation). It is achieved by the use of message authentication codes.

Security policies should be defined considering the definition of different levels of access control, including the possibility of authorizing anonymous accesses to certain sensor nodes or resources.

3.4.1.8 General communications performance

General communication performance includes all the requirements that affect the capability of data transmission in the network. Also, as nodes have limited resources and are many times located in places with difficult access, energy issues are of utmost importance. Not only is necessary to know the energy levels of each node but also to calculate the predicted node and network lifetimes.

The main requirements of this group are the delay tolerance, loss tolerance, capacity, reliability, energy efficiency, criticality and fault tolerance.

The delay tolerance is one of the main issues when considering time-critical applications [48] such as in industrial or health environments. When that tolerance is surpassed the risks for the health of a patient or to a industrial installation increases rapidly.

Loss tolerance is also an important issue in every network. Each application has its own bound for packet loss. It can be more tolerated in agriculture applications but very strict in industrial ones.

The overall capacity of the network to transmit information is also important for applications that involve heavy traffic. Additionally, it is also important to control the percentage of the capacity in use at the moment.

Network reliability assesses how the network should comply with the delivery of packets. More than just looking at specific loss values, reliability expresses the trust that the application has that its data is delivered or received. Reliability can decrease due to multiple factors such as harsh environments, transmission errors, collisions and network congestion. It is an essential requirement in industrial scenarios [49] and important in most scenarios [50].

Energy is a vital, but also limited, resource in WSNs. Even in the cases when nodes have their own sources of energy (e.g. solar panels), applications must know if nodes have the adequate energy levels to comply with their needs. Also, it is essential to assess its efficiency [48], if the amount of energy that is being spent is the expected for the work done.

By including critical applications and environments in the scope of WSNs, the requirement for different priority packets raises [44]. To assess if priority packets are having the expected treatment by the network, some specific bounds must be specified for the transmission of these packets.

As with any other types of networks, fault tolerance to nodes or data transmission failure, is also an important requirement in a WSN, specially when dealing with critical applications as in industrial scenarios [49]. Typical fault tolerant requirements for WSNs are the number of active nodes or the average time to detect a node failure.

3.4.1.9 Deployment

Deployment group focuses on the different deployments possible - random or deterministic. In a random deployment there is no control over the final position of the nodes. In a deterministic deployment nodes are put in specific positions that resulted from a specific plan. Each of the deployment options implies specific needs (e.g. self configuration in random placed networks) and involves the use of a set of specific tools.

3.4.1.10 Coverage

The Coverage group addresses the guarantees of the network in what respects the cover of the point of interest. Coverage can be divided in two groups: *space* and *time*.

Space coverage relates to the physical coverage of the environment/event/phenomenon by the WSN. It includes requirements such as the ability of the network to detect objects and events in the sensor field, and the network exposure to the same events (i.e., the network ability to observe an object/event moving on an arbitrary path over a period of time by using the existing sensor nodes).

Time coverage deals with the amount of time the environment/event/phenomenon is effectively addressed by the WSN and includes requirements such as the lifetime and the duty-cycle of the nodes. Lifetime can be divided in network lifetime and node lifetime. Network lifetime is the maximum time that the network fulfils its objectives (such as the time until the first node fails or the time when the network stops providing the application with the necessary data from the sensors) – commonly referred as the network operational time. Node lifetime is the time until the node stops providing the application with the necessary data. The node's duty cycle is the percentage of time a node is active, considering the whole operational time.

3.4.1.11 Network Management considerations

An additional factor that influences the QoSensing requirements of a WSN relates to the implicit needs of network management. To address network management, ISO identified five functional areas [51]: *fault management*, *configuration management*, *accounting management*, *performance management* and *security management* (FCAPS). Fault management deals with the functions used to detect, isolate, correct and log faults that occur in the network. Configuration management include functions to gather, modify and store configurations from network devices. Accounting management tracks the use of resources so that they can be billed. Performance management collects and analyses performance data so that resources and communications can be evaluated. Finally, Security management deals with the security services available and on the report of security events. New WSNs must include the same functional areas but, due to their specific properties, have to extend the existing ones and add some new. As an example, wired performance management must give place to a WSN performance management that includes network coverage and connectivity, and new functional areas like topology management and energy management must be created.

As the needs implied by the network management of WSNs do not always configure groups of specific requirements by themselves, it was decided to include them in the existing taxa (taxonomy groups), especially in the Security and Privacy, and in the Performance group. Configuration management needs are included in Application specific needs (in this case a configuration application).

3.4.2 Complete taxonomic tree of requirements proposal

After having described the main groups of the taxonomy, this sub-section presents the complete schema of the taxonomy proposal that includes not only the groups already referred but also the sub-groups discussed for each main group. The taxonomic tree of a WSNs QoSensing requirements proposed includes all the QoSensing requirement groups found needed to characterize each WSN requirement and provide for a WSN with controlled performance, used for sensing or acting in the environment. The classification addresses the network perspective, but also considers the performance of the information obtained and transmitted by the network, according to what was already explained in the previous section.

The classification is presented using a user/application perspective in the sense that it addresses the requirements of the user/application from the WSN, and by considering that the global performance of the WSN (its QoSensing) translates into application performance and on user satisfaction. The classification focus on the user/application needs, not on how it is accomplished by the network.

The objective is that this classification will not only help to, in a next phase, create the appropriate metrics that permit the effective quantification of the requirements, but will also allow that the applications themselves may specify their requirements based in a common classification.

An initial version of our proposal of a taxonomic tree of requirements of a WSN was discussed in the context of ISO/IEC JTC1 - *Study Group on Sensor Networks* (SGSN). This group was focused in "fulfilling the Terms of Reference from ISO/IEC JTC 1, which is the study of Sensor Networks" [27]. A detailed version was accepted for inclusion in the final report ([27]) and a smaller version was published in [36]. This version was later revised and updated, what led to the final version of the proposed taxonomy that is depicted in Fig. 3-4. (Performance branch is highlighted because it will be specifically addressed in Section 4.5).

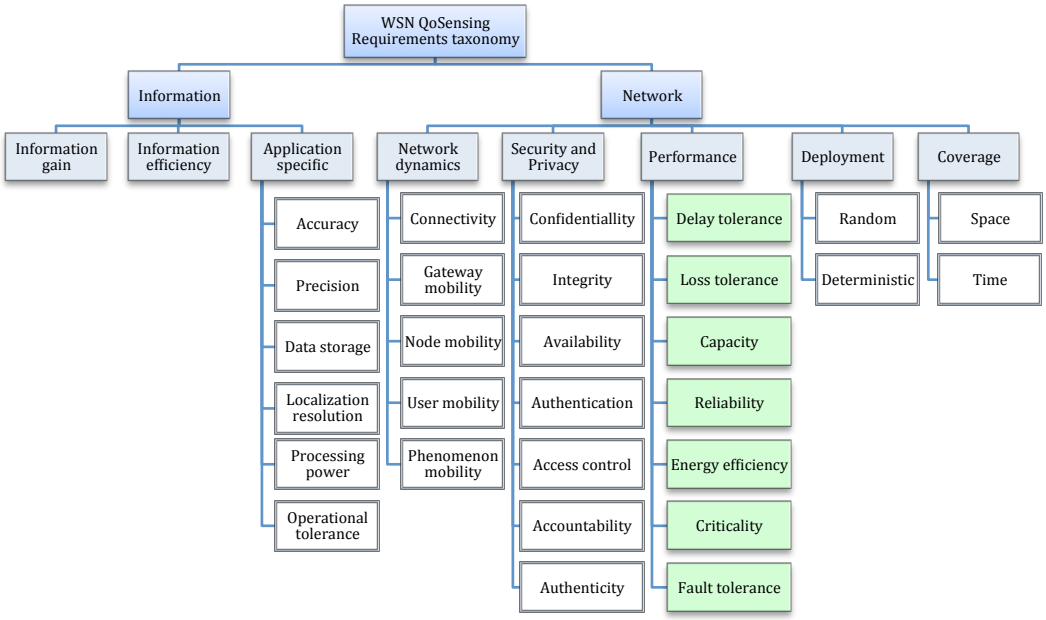


Fig. 3-4 - Final version of a taxonomic tree of WSNs QoSensing requirements.

3.4.3 Considerations about the proposed taxonomy

The taxonomy proposed was a response to the lack of alternatives found when trying to classify the requirements of the GINSEG scenarios and its QoSensing needs, specially in an application/user perspective, as explained in Section 3.2. The proposed taxonomy includes the necessary dimensions that were found necessary to characterize the QoSensing requirements of any WSN, including WSNs with controlled performance. In order to evaluate these requirements, metrics should be created from the available parameters. However, not all requirements must be translated to metrics to be monitored by the network, as some depend simply on the hardware, on the type of sensors used or in the form of deployment. Also, it is not essential that all nodes in a network have the same requirements.

Nevertheless, this taxonomy does not include all the possible requirements of a WSN. Analysing the related work of this chapter one can find taxonomies that include dimensions such as “Cost, Size, Resources and Energy” [31], “Users” (e.g. single, competitive) [34] and “Goal” (e.g. sense-only, sense-and-react) [38]. While these dimensions may be considered requirements and are useful to understand the financial effort to create the WSN, or in the design of the programming approaches to those networks, they are not relevant when assessing the QoSensing that the network renders to the user/application that is the final recipient of the data. The contrary is not always true, with QoSensing requirements influencing costs and the choice of the programming approaches to be taken. Also, requirements as scalability, a very important issue when evaluating a network to be used with a specific application, while not directly present in the proposed taxonomy, may be specified by considering different existing requirements

at the same time, as space and time coverage, delay, reliability and information efficiency.

While not specifically designed for the characterization of protocols or algorithms, the proposed taxonomic tree may also prove to be useful in those domains. As an example, if designing a data aggregation algorithm, its efficiency may be expressed in requirements such as time coverage, data accuracy and delay.

3.5 APPLYING THE CLASSIFICATIONS

In this section the classification of applications and the taxonomy proposed are applied to 4 different kinds of scenarios: industrial, target tracking for military use, wildlife monitoring and precision agriculture. While the first two are examples of networks with strict performance boundaries and critical applications, the last two are common applications of WSNs found in literature that represent non-critical environments. For each of the scenarios the applications used were respectively GINSENG ([5] [21]), VigilNet ([52]), ZebraNet project ([40]) and a potato crop project ([53]). In order to understand and analyse the specific GINSENG performance requirements, both the proposed WSN application classification and taxonomy were applied to the scenarios presented in Section 2.5.2, which were used in the definition of the project: (A) Production Monitoring, (B) Production Control, (C) Production Monitoring and Control, (D) Pipeline Leak Detection and (E) Personal Safety.

Table 3-1 presents the classification of applications using the classification proposed in Fig. 3-1.

TABLE 3-1 – EXAMPLES OF WSNS APPLICATIONS USING THE CLASSIFICATION OF FIG. 3-1

Application	Deployment dynamics	Deployment strategy	Control scheme	Data delivery model	Time constraints
GINSENG A	Static	Deterministic	Open-loop	Continuous	Time-constrained
GINSENG B	Static	Deterministic	Closed-loop	Continuous Event-driven	Time-constrained
GINSENG C	Static	Deterministic	Closed-loop	Continuous Event-driven	Time-constrained
GINSENG D	Static	Deterministic	Closed-loop	Continuous	Time-constrained
GINSENG E	Hybrid	Deterministic	Open-loop	Event-driven	Time-constrained
Wildlife monitoring	Mobile	Random	Open-loop	Continuous	No time-constrained
Military Target Tracking	Static	Deterministic	Open-loop	Event-driven	Time-constrained
Precision agriculture	Static	Deterministic	Closed-loop	Continuous/ Query-driven	No time-constrained

Next table shows the use of the proposed taxonomy of QoSensing requirements in the classification of the specific requirements of example WSNs applications and scenarios. Each one of the requirements found is mapped to its specific taxon.

TABLE 3-2 – MAPPING EXAMPLES TO TAXONOMY IN FIG. 3-4

Scenario	Classification	
GINSENG A	Information gain	Frequency of packet generation >5 s; packets < 10 bytes
	Application specific	On node processing for filtering of traffic
	Network dynamics	Static
	Security and Privacy	Authentication; integrity; confidentiality
	Performance	Delay < 3 s; low packet loss
	Deployment	Deterministic
	Coverage	Lifetime >6 months
GINSENG B	Information gain	Frequency of packet generation [2, 5] s; packets < 10 bytes
	Application specific	On node processing for filtering of traffic
	Network dynamics	Static
	Security and Privacy	Authentication; integrity; non-repudiation; confidentiality
	Performance	Delay < 2 s (upstream),<1 s (downstream); no packet loss
	Deployment	Deterministic
	Coverage	Lifetime >6 months
GINSENG C	Information gain	Frequency of packet generation [1, 2] s; packets < 10 bytes
	Application specific	On node processing for filtering of traffic
	Network dynamics	Static
	Security and Privacy	Authentication; integrity; non-repudiation; confidentiality
	Performance	RTT<2 s; no packet loss
	Deployment	Deterministic
	Coverage	Lifetime >6 months
GINSENG D	Information gain	Frequency of packet generation >1 s; packets < 10 bytes
	Application specific	On node processing for filtering of traffic
	Network dynamics	Static
	Security and Privacy	Authentication; integrity; non-repudiation; confidentiality
	Performance	RTT<2s; no packet loss
	Deployment	Deterministic
	Coverage	Lifetime >6 months
GINSENG E	Information gain	Frequency of packet generation >5 s; packets < 10 bytes
	Application specific	On node processing for filtering of traffic
	Network dynamics	Sensor mobility (workers)
	Security and Privacy	Authentication; integrity; non-repudiation; confidentiality
	Performance	RTT<5s; no packet loss
	Deployment	Deterministic
	Coverage	Lifetime >6 months
Military target tracking (VigilNet)	Info gain	Very low false negative rate; Low false positive rate
	Info efficiency	Time-constrained sensing and classification to avoid spending energy
	Application specific	Stealthiness; precise location estimation
	Network dynamics	Static
	Performance	Real-time (low latency); energy-efficient
	Deployment	Deterministic
	Coverage	Lifetime: between days and months depending on the mission
Wildlife monitoring (ZebraNet)	Information gain	Frequency of reading varies from a TDMA epoch to 2 weeks depending on parameter
	Application specific	Precise location
	Network dynamics	Static
	Performance	Low packet loss (uses TDMA)
	Deployment	Deterministic
	Coverage	Space- all field (max 10m apart); lifetime >4 months (crop time)
Precision agriculture (Potato crop)	Information gain	3 minutes logs per hour
	Application specific	Precise location; 640 KB storage; data homing rate (information sent to sink) close to 100%
	Network dynamics	Mobile
	Performance	Close to 100% reliability
	Deployment	Random
	Coverage	Node range from 100 m (small range) to 8 km (longer range)

The previous table shows the QoSensing requirements of each of the examples selected. For some examples no requirements exist for all taxonomic groups, as the application observed does not use them.

3.6 CONTRIBUTIONS TO GINSENG PROJECT

After a detailed analysis of the GINSENG scenarios using the taxonomy, presented in the previous section, and discussion with other members of the project, a set of performance requirements was chosen and proposed as the most important, and proposed for the project. These performance requirements were divided into two priority groups as shown in the next table.

TABLE 3-3 – PRIORITY PERFORMANCE REQUIREMENTS ([21])

1st Priority Functional Requirements
Message Delay
Message Delivery Reliability
2nd Priority non-Functional Requirements
Fault Tolerance
Energy Efficiency
Security
Limited Mobility

The first priority group deals with performance issues that were found to be the most relevant: delay and delivery reliability. Delay is of crucial importance in critical industrial environments where the response time to an event has strict time boundaries. Also, by controlling delay, the existence of high priority traffic (e.g. alarms from sensor nodes, worker health problems alerts while working in hazardous areas of the refinery) is enabled. Message delivery reliability provides guarantee of accuracy and delivery of data, which are also fundamental. The second priority group identifies additional areas of interest, such as fault tolerance (including connectivity), energy efficiency, security and limited mobility, which are important, but not necessarily essential, to the operation of a network with controlled performance. This proposal was initially included in Section 5 of project deliverable D1.1 – "GINSENG Architecture, Scenarios and Quality of Service Measures" [25] and more detailed in Sections 2.5.1 and 2.5.2 of project deliverable D1.3 – "Final GINSENG Architecture, Scenarios and Quality of Service Measures" [21], together with an initial version of the taxonomy and a classification of the GINSENG scenarios.

3.7 CHAPTER SUMMARY

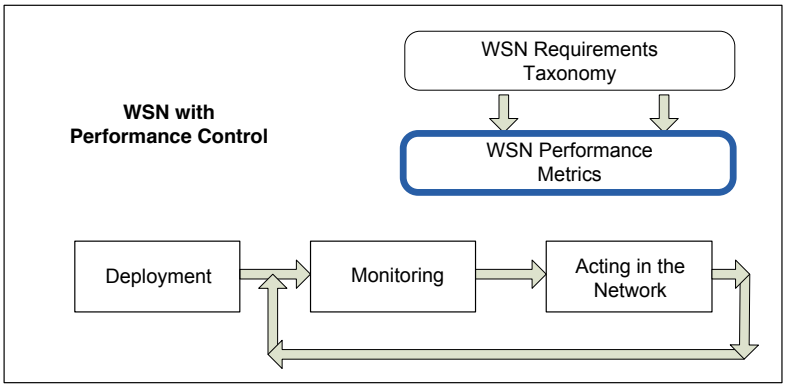
This chapter was the result of an effort to characterize and describe the QoSensing requirements of a WSN, specially considering WSNs with controlled performance, while taking into account the specificities of these networks. To accomplish it, a new classification of WSN application scenarios, that also includes critical environments, was

proposed. This new classification was found essential, as WSNs requirements are heavily dependent on the applications they run. Next, a proposal of a new taxonomic classification of requirements for WSNs, including those with performance control, was presented. Its objective was to characterize WSNs QoSensing needs, both in the information as in the network planes. The creation of a reference classification is the first step in the evaluation of the global performance of WSNs, as it establishes the requirements that need to be addressed. Next chapter will focus in the creation of metrics that allow for the measurement and evaluation of part of the requirements found.

Chapter

4

WSN Metrics Proposal



In the previous chapter the QoSensing requirements of a WSN were classified according to a new taxonomy. However, to evaluate each group of requirements found, the taxonomy must now be transformed into a set of usable metrics.

In this chapter, the Network performance branch of the taxonomy will be addressed and specific metrics will be chosen and created to evaluate it. To accomplish it, different types of metrics and their advantages and disadvantages when applied to WSNs, are analysed, and a general set of specific metrics of different types is proposed. Next, each of the different types of metrics is evaluated in order to assess which type is more efficient in giving indicators on the global Network performance QoSensing of a WSN. Based on the results obtained, and in the minimum requirements of a WSN in industrial scenarios, a framework to assess the special needs of a WSN monitoring system in generic industrial scenarios is proposed. Finally, specific metrics from the initial set proposed are used to dynamically control the working channel of a network, by acting in the network.

4.1 INTRODUCTION

A metric is a fully specified value resulting from the measure of some quantifiable reality. A metric must be carefully specified, without ambiguities, and should be defined in terms of standard units of measurement. It is also possible to define a metric just in terms of other metrics. These metrics are called 'derived metrics'.

The measurement of performance in WSNs as it is done today, is mostly an extension of the measurement of performance provided for wired networks, with some additional care for energy waste. However, WSNs are very different from typical wired networks, or even the common Wi-Fi networks, requiring a different performance measurement and presenting several restrictions. As can be easily perceived, WSNs performance not only comprises the common delay, delay variation, bandwidth and packet loss, for each individual node, but also a set of indicators that arise from the particular nature of WSNs and its applications. In this group one can find metrics such as coverage and spent energy, or some more specific like target tracking error. Also, the specific restrictions that WSNs have, either in hardware and software, translate into lower computational power, small memory and severe energy limitations, to name some. These restrictions pose some new challenges in the performance assessment of this particular type of networks, such as the need of simple metrics and additional care for its collection. Furthermore, many of these networks are constituted by several nodes, many of them redundant in order to achieve global network reliability. The information collected is many times also redundant, as are its individual performance measurements. Even the typical end-to-end performance measurement scenario may not always be usable in WSNs, where there are events whose influence spans to more than one node and where some performance issues are dependent from a group of nodes. Because of all the facts referred, a global understanding of the performance of a WSN is yet to be achieved.

The goal of this chapter is to find a suitable group of metrics, adapted to WSN, that can evaluate the Network performance branch of the taxonomy presented, the one that deals with the requirements that affect the capability of data transmission in the network, and propose a small set to be used in generic industrial scenarios.

4.2 RELATED WORK

Performance in WSNs has become an important issue when an effort was made to use the flexibility of these networks in critical, or at least performance sensible, environments. The lack of guarantees of performance from typical WSNs led to the development of specific architectures and standards that tried to compensate the restrictions of these networks. Specifically, the efforts to bring wireless sensors and actuators to industrial scenarios resulted in architectures such as the ISA100.11a [17], WirelessHart [18] or GINSENG [5]. By defining specific algorithms and protocols, together with a specific architecture, they provide the basis for WSNs industrial performance needs and facilitate

the achievement of specific QoSensing objectives. Nevertheless, to assure that the QoSensing needed is met during operation, a constant performance monitoring is necessary.

Although an exhaustive study has been made for wired networks, resulting in a Framework for the IP performance evaluation, including the use of composition and aggregation of individual metrics over time and space (RFC2330, RFC5835) [6][54], the same has not yet been done considering all the specificities of WSNs. Due to their nature, WSNs have a broad set of requirements (such as energy or coverage), the focus is less on each device and more in the global network (as variations in individual measurements are typical) and their measurements are not necessarily end-to-end.

However, the subject has been under intensive research in recent years, led by the increasing interest of applying the flexibility of WSNs to performance demanding scenarios.

Specific QoS in WSNs is analysed by Chen and Varshney in [4], in terms of end-to-end, interactivity, delay tolerance and criticality by application delivery model. They also introduce collective QoS parameters as a response to the specificities of the measurement of QoS in WSNs. Namely, collective latency, collective packet loss, collective bandwidth and information throughput are defined to deal with non-end-to-end WSN QoS parameters.

Some guidelines to WSN performance benchmarking, in search for a common methodology to collect, compare and evaluate different optimization methods, are proposed by Martinovic et al. in [55]. They also propose metrics such as energy consumption, network lifetime, average delivery ratio, average packet delay, average overhead, total data aggregation, standard deviation of remaining energy in nodes, throughput, average packet journey length, response time and sampling frequency. None of these metrics are tested in the paper.

Specific studies focusing on particular aspects of WSNs performance were also presented. As examples, A.Sen et al. in [56] deal with fault tolerance, introducing a new concept of region-based connectivity, N.Ahmed et al. [57] analyse WSNs tracking systems using estimation of the tracking error (distance between the predicted and the actual location of the target) and computation time as metrics, and Dietrich et al. [58] use different metrics to create a new definition of network lifetime that also indicates the performance degradation of the WSN.

The analysis of performance using nodes local storage and computation capabilities, instead of sending all information to the sink, is discussed by O'Donovan et al. in [59].

Aggregation in WSNs has been subject of many studies, focused on general data, and considering the specificities of these networks. Examples of these studies are [60] and [61].

A framework for WSNs performance monitoring was presented by the author of this thesis in [62]. It presents a selected set of metrics that intend to assess the communication performance of WSNs and provide a common ground for WSNs performance comparison. Collective network metrics are proposed to aid in the measuring of performance. This work is fully developed in this chapter.

In [63] Lingyun and Xingchao propose a multi-criteria performance evaluation bringing together traditional networks common performance criteria such as network delay, loss rate, bandwidth and traffic, and also some new criteria such as energy consumption, network lifetime, network coverage degree, connectivity degree and fault tolerance. They also present a synthetic performance evaluation method based on a modifiable criteria weight and regarding the network path as the basic performance evaluation unit.

Some metrics use clock synchronization to be able to measure its values. A survey of time synchronization mechanisms for WSN is made in [64] and [65]. Sichitiu and Veerarittiphan [66] and Sommer and Wattenhofer [67] also present some interesting protocols.

In spite of some efforts in creating an universal performance framework for WSNs performance measurement, most of the work found on WSNs performance related literature is very limited and does not deal with the specificities of WSNs with controlled performance in broader terms. Most of the works presented address specific measurements that apply to the particular subject in focus, such as protocol comparisons, or independently address specific aspects of performance, such as coverage, energy, fault tolerance, lifetime or connectivity. The fact that most of the current research on WSN performance targets specific aspects and not a global framework, does not permit a global view and perception of the global performance provided by the WSN. Moreover, none of the approaches is indicated to assess performance in critical scenarios like industrial plants.

4.3 AN OVERVIEW OF IPPM METRICS

The issue of performance in traditional wired computer networks, with little restrictions in what respects computational power, memory or energy, has been deeply investigated. In IP networks, IETF's *IP Performance Metrics* (IPPM) Working Group of the Transport Area, proposed a framework for IP performance evaluation (RFC2330) [6]. The motivation to its development was to give users and providers of Internet service a common understanding and framework for measuring the performance offered and

obtained. The metrics are designed for the use of network operators, independent testing groups, or end users, and represent an unbiased quantitative measure of performance.

The IPPM proposal is one of the most comprehensive approaches to the performance of computer networks; in this case, IP networks. While this framework does not apply directly to the performance in WSNs, a study of its characteristics is done here as a guideline approach.

4.3.1 Introduction of IPPM

Metrics to measure performance under IP networks have been addressed by IETF's IPPM working group of the Transport Area and published as RFCs under the framework proposed in the Framework for IP Performance Metrics (RFC2330, RFC7312) [6] [68].

The metrics can be divided in two main groups. The first group contains the basic metrics to measure IP performance and is constituted by the IPPM Metrics for Measuring Connectivity (RFC2678) [69], One-way Delay Metrics (RFC2679) [70], One-way Packet Loss Metrics (RFC2680) [71], Round-trip Delay Metrics (RFC2681) [72] and Round-trip Packet Loss (RFC6673) [73]. The second group deals with specific Internet behaviour and specifies new concepts of performance. It is constituted by One-way Loss Pattern Sample Metrics (RFC3357) [74], IP Packet Delay Variation Metric (RFC3393) [75], IPPM Metrics for Periodic Streams (RFC3432) [76], Packet Reordering Metrics (RFC4737) [77], Network Bulk Transport Capacity Metrics (RFC3148) [78], Network Link Capacity Metrics (RFC5136) [79] and One-way Packet Duplication Metric (RFC5560) [80]. Also, the composition, decomposition and aggregation of individual metrics over time and space are also presented (RFC2330, RFC5835) [6] [54]. Additionally RFC5644 [81] introduces metrics for measuring segments of a path and metrics between a source and many destinations.

In the context of IP performance, a metric assesses the performance or reliability of the operational Internet, giving users and providers a common unbiased language to characterize the service.

IP metrics were defined using an analytically and empirically approach. Analytical metrics are based in a common analytical framework of concepts developed by the Internet engineering community [6][82]. Example of an analytically defined metric [6]: "propagation time of a link - the time, in seconds, required by a single bit to travel from the output port on one Internet host across a single link to another Internet host".

If a metric is too complex to discuss analytically but still very important for practical measurement, a reference methodology for measuring it can be described. These are the empirically specified metrics. Nevertheless, the metric should also be specified analytically to the extent possible, although in an incomplete manner.

4.3.2 Basic Metrics

Basic metrics are metrics such as connectivity or delay that capture the raw performance of the Internet, and that can be measured by analysing a single packet. Table 4-1 presents the metrics proposed by IPPM.

TABLE 4-1 – BASIC IPPM METRICS

Metrics	Description
Connectivity (RFC2678) [69]	Using a Boolean unit, connectivity measures if two hosts can reach each other. *Instantaneous one-way connectivity*, *Instantaneous bidirectional connectivity*, *One-way connectivity*, *Two-way connectivity* and *Two-way Temporal Connectivity* are defined.
One-way Delay Metrics (RFC2679) [70]	Measures the time, using a real number of seconds, that a packet spends travelling between two hosts. It includes the propagation and transmission time of the packet. For the measurement of this metric both clocks must be synchronized.
One-way Packet Loss Metrics (RFC2680) [71]	Measures if a packet sent from source to destination arrived in a reasonable (defined before) amount of time. The metric unit is 0 if the packet was not lost and 1 if it was lost.
Round-trip Delay Metrics (RFC2681) [72]	Measures the time a packet takes from source to destination and then to source again. On reception by the initial sender of the packet, the subtraction of the final timestamp and the initial one gives the round-trip delay. The metric unit used is a real number of seconds.
Round-trip Packet Loss (RFC6673) [73]	As many user applications and transport protocols use two-way (round-trip) communications, the measurements of Internet round-trip packet loss performance provide a basis to infer application performance. It measures if a packet sent from source to destination and its response to the original source arrived in a reasonable (defined before) amount of time, and returns a Boolean.

4.3.3 Internet Behaviour Metrics

In order to detect specific network behaviours in the Internet, which affect users and operators, additional metrics were created. These metrics constitute the group of Internet behaviour metrics and are presented in Table 4-2.

TABLE 4-2 – IPPM INTERNET BEHAVIOUR METRICS

Metrics	Description
One-way Loss Pattern Sample Metrics (RFC3357) [74]	*Loss distance* (the distance, in number of packets, from two successively lost packets) and *Loss period* (moment in the transmission where losses occur) were created from the knowledge that for the same loss rate, different loss distributions could result in different performance observed by users. This happens in real-time applications and for non-real-time applications that use an adaptive protocol such as *Transmission Control Protocol* (TCP). The unit used in both cases is expressed as an integer.
IP Packet Delay Variation Metric (RFC3393) [75]	The *IP Packet Delay Variation* (also known as jitter, now deprecated) measures the variation in delay of packets across an Internet path. To make the measurements, the clocks used by the two hosts do not have to be synchronized. This metric helps to size the buffers along the network path and is specially indicated for streaming applications. The metric unit used is the real number of seconds.
IPPM Metrics for Periodic Streams (RFC3432) [76]	This type of metrics is similar to previously presented basic metrics but uses periodic streams of data to make the measurements. This group defines metrics such as *One-way delay Periodic Stream*, *Packets not received*, *Corrupt packet headers*, *Duplicate packets*, *Variation of delay* (between consecutive packets), *Average delay* and *Maximum delay*.
Packet Reordering Metrics (RFC4737) [77]	Packet reordering metrics evaluate whether a network maintains packet order on a packet-by-packet basis, as sent by the source. Packets are ordered if their sequence number (a unique consecutive integer that establishes the source sequence) increases with each new arrival and there are no backward steps.
Network Bulk Transport Capacity Metrics (RFC3148) [78]	*Bulk Transport Capacity* (BTC) is the measure of a network's ability to transfer significant quantities of data with a single congestion-aware transport connection (e.g. TCP). In practical terms can be measured as the expected long-term average data rate (bits per second) of a single ideal TCP implementation over a specific path (BTC = data_sent / elapsed_time). The data sent refers to bits of actual data, not including header bits, and only includes the unique number of bits transmitted (retransmissions of data are not counted).
Network Link Capacity Metrics (RFC5136) [79]	To avoid measuring information-carrying capacity, as it varies with the protocol layer, is not necessarily fixed and depends on the type of packets used (as different packets may have different treatments in the network), metrics such as *IP-layer link capacity, IP-layer Path Capacity, IP-layer Link Usage, IP-layer Link Utilization* were created. This metrics deal not only with unique application data, but also with the IP header and retransmitted data.
One-way Packet Duplication Metric (RFC5560) [80]	Two packets are considered identical if both contain identical information fields (sent from the same source and containing the same information). However, this does not mean that all bits in the packet are the same (e.g. TTL value in the packet may be different).

4.3.4 Composition and aggregation of metrics in time and space

The analysis of elementary measurements is not enough to enable the network operators to understand completely the network's behaviour. Most of the times, measurements need to be post-processed so that specific performance data, which cannot be obtained from

individual data, could be derived. This processing can take the form of a composition and/or aggregation of individual data. These mechanisms provide for lower overhead in terms of storage for the results and on traffic in the network, and an easier way to detect trend changes or anomalies in the network, enabling performance estimation based on performance data from smaller parts of the network. However, a smaller amount of error is expected, in addition to the existing error inherent to the measures themselves.

The concept of composing metrics in time and space was already an item in RFC2330 that originally defined the Framework for IP Performance Metrics. The informational RFC5835, although not a standard, further expands the notion of metric composition and describes the mechanisms of composition and aggregation in generic terms. While RFC2330 uses the terms spatial and temporal composition, RFC5835 changed them to spatial and temporal aggregation, not considering the exploiting of time correlation that certain metrics can exhibit (that were present in RFC2330).

As defined in RFC5835, spatial aggregation of a metric implies that the value of the metric along a path *P* is related and can be obtained by knowing the value of the same metric in all of the sub-paths that compose *P*. Space aggregation is generally useful to obtain a summary view of the behaviour of large network portions. As an example, the delay in a path *P* could be obtained, with a small error, by the sum of the delay of every sub-path of *P*. Temporal aggregation of a metric implies that the value of the metric along a path in a time interval *T* is related and can be obtained if the values of the same metric in all the sub-intervals of *T* are known. Time aggregation is useful to reduce the amount of data that has to be stored, to detect cycles or trends easier and to detect anomalies in the network. Extrapolation of a time interval based in observed behaviour is also addressed in RFC2330. As an example, the packet loss in a specific path during a time interval *T* would be the sum of the losses in each of the sub-interval of *T*. While in the examples referred only the sum function was presented, every aggregation function can be used.

Composed metrics might themselves be subject to further steps of composition or aggregation, in higher-order compositions.

4.3.5 Spatial and Multicast metrics

IPPM, in RFC5644 [81], extended the coverage from end-to-end performance between two points, to measurements involving multiple measurement points. It defines spatial metrics for measuring the performance from source to a destination path, and metrics for measuring the performance between a source and many destinations in multiparty communications, which involve more than one measurement collection point (e.g., a multicast tree). All these metrics are based on one-way end-to-end metrics and follow the IPPM framework in RFC2330 [6]. It defines vectors that contain the values of a metric along a path and matrixes that group vectors along a time interval. The vector gives

information over the dimension of space and the matrix represents the network performance in both space and time. One-to-group metrics are also defined using vectors, measuring the one-way performance between one sender and N receivers. Based on these metrics, statistics are also defined to characterize single receiver performance, group performance, and relative performance.

4.3.6 Some considerations

Performance of an application depends mostly on the performance of one direction. Symmetry not always exists and can be very different on QoS enabled networks where provisioning may differ in both directions. Also, asymmetric queuing may also result in asymmetric performance even in symmetric paths. [70]

Measurements

A typical measurement scenario is shown in Fig. 4-1. The scenario has one source, one destination and 2 different *Measurement Points* (MP) that may be included in source and destination or have separate equipment. The latter should be used if using source or destination can significantly affect the delay performance to be measured. *MP(Source)* should be placed close to the egress point of packets from source. *MP(Destination)* should be placed close to the ingress point of packets for destination. MPs should be close enough to source or destination so that the performance measured may accurately reflect the existing performance [76].

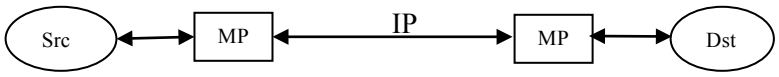


Fig. 4-1 - Measurement setup [76] **(Src=Source; Dst=Destination; MP=Measurement Point).**

Measurements by type

Measurements can be one-way, paired one-way, or round-trip. One-way measurements perform a single one-way measurement that only yields information on the network behaviour in one direction. Paired one-way measurement involves measurements from source to destination and from destination to source. This type of measurement enables the measurement of both directions of transmission and gives a more accurately knowledge of the network than a round-trip measurement as each direction may have different characteristics and performance. Round-trip measures the total path from source to destination and back to source. By making the measurements in the source it avoids the need of clock synchronization between the hosts.

Measurements by protocol layer

The five TCP/IP protocol layers - physical, link, network, transport and application – may provide different performance measurements [76]. In a user application point of view, the

focus is on the transport layer as it provides the overall performance of the layers bellow and translates the network contribution to the QoS perceived by the user. However, to separate the contributions of each layer, separate measurements can be made. To separate the quality contribution of the operating system or of the codecs in video or voice data, measurements may be done in lower layers. As an example, throughput may be measured at transport layer, packets counted at network layer, bit error rates measured at the link layer.

Active and Passive measurements

The metrics developed under the IPPM framework involve active measurements, i.e., in order to measure the metrics dedicated traffic must be produced. However, it is also possible to make measurements under a passive context, by using the existing traffic. [6]

Measurements protocols

RFC3763 [83], an informational RFC, defines the requirements for a one-way active measurement protocol that allows users to do measurements using devices from different vendors at both ends, with coherent results. With a standard technique to collect one-way measurements, metrics could then be collected across more paths using open one-way active measurement beacons, and be as easy to obtain as current round-trip time is measured using *Internet Control Message Protocol* (ICMP) based tools like ping. A standard, *One-Way Active Measurement Protocol* (OWAMP), was proposed in [83] [84], which measures unidirectional characteristics. Since this standard protocol does not accommodate round-trip or two-way measurements, a new standard, *Two-Way Active Measurement Protocol* (TWAMP) was proposed in [85]. A round-trip packet loss metric specified according to the RFC 2330 framework was later proposed in [73].

4.4 METRICS FOR WSNS WITH CONTROLLED PERFORMANCE

QoS in WSNs can be monitored using traditional metrics like delay, loss, or bandwidth, collected from all the individual nodes in the network, each analysed individually. However, these metrics are not enough to measure the WSN QoSensing or even to assess its global performance in an efficient way. Furthermore, measuring these metrics, as proposed by IPPM, raises some problems and does not give a correct overview of the global performance of the network as it is perceived by common WSN applications.

Problems start to arise from the special nature of WSNs. WSN nodes have scarce resources, such as a reduced calculation power, a small amount of memory and limited energy. The delivery of all the metric values from each node to the sink, which most of the times is many hops away, depletes energy from upper level nodes (upstream nodes) and implies a waste of bandwidth, so necessary in networks that may have a large number of nodes or require a long network life. So, metrics collected from nodes must be optimized, simple to calculate and to send, wasting the minimum amount of resources

possible. Also, if possible, the sink should infer the metrics, without a special delivery of performance packets. At the same time, values collected from wireless nodes are subject to variations and communication errors that lower the quality of data. This happens because nodes are many times placed in areas with difficult human access, subject to interference and hazards. As a result, values are more reliable when not individually observed and when individual values are not as highly valuated as the group from which they are part of. This happens not only to values of sensed data, but also to performance measurements data. As an example, a high value of packet loss may not indicate a network or node malfunction but just an unexpected interference that affected part of the network.

Also, the use of traditional metrics, while useful, does not give a clear insight of what the QoSensing of the WSN is. As these networks are very dynamic, subject to many kinds of hazards due to their nature, items such as energy efficiency, fault tolerance and reliability, are also part of their performance. At the same time, due to the applications used, other issues such as accuracy, precision, coverage may also be needed to assess the global QoSensing of the network.

Next, an approach to different types of metrics that can be used in WSN, together with their strengths and weaknesses, and an approach to its definition considering WSN restrictions, are presented.

4.4.1 Metric types: Individual, collective and composed

In a computer network, the parameters from which the metrics are calculated can be either individual or collective. Individual parameters are related to a single node, while collective parameters are the result from the composition of parameters of a group of nodes. The use of individual parameters results in individual metrics while the use of parameters from different nodes results in collective metrics.

Individual metrics relate to only one node and are transmitted end-to-end. These metrics are essential when debugging a specific node or when assessing its specific performance. However they do not give any insight about the behaviour of the network as a whole. Furthermore, constant tracking of individual metrics (that includes calculation and transmission) leads to the depletion of wireless nodes, both the ones where the metric is calculated as the ones that have to transmit, hop by hop, the information to the sink. As an example, the calculation of one-way node packet loss at each node does not involve hard calculations but a constant report to the sink, which multiplied by the number of nodes in the network, does create an avoidable waste of resources, such as energy or bandwidth. Moreover, the loss of packets in a specific moment does not necessarily imply any problem in the network or in one specific node, it may just be a momentary interference so common in most of the WSNs deployment scenarios. While individual metrics may be sent along with data packets, therefore reducing its impact in the network (as it avoids the

need to send extra packets), this is not always needed and its avoidance may save some additional energy.

When calculating metrics based on collective parameters, which we will address from now on as collective metrics, the reality observed is not individual but involves a predefined part of the network or even the all network. The use of WSNs collective parameters was initially proposed in [4] to deal with WSNs specificities and its calculation involved the use of values from more than one node, considering the same event. As an example, in a data-gathering application, collective packet loss would be the sum of individual packet losses from all the sensing nodes that sent data related to a specific event (by event is meant an occurrence of a phenomenon that is sensed), and collective delay the difference between the time the data was obtained and the time when the last packet concerning the event, from all targeted nodes, arrived at the sink. This definition can now be extended considering parameters obtained from all (or a specific group of) network nodes, instead of a specific event and the nodes involved. Considering this context, it was proposed by the author of this thesis [62] to divide collective parameters in collective event parameters and collective network parameters, to create collective event metrics and collective network metrics. Using the same example as before, packet loss, the collective metric "collective network loss" gives the number of packets lost in the entire network and its values should remain statistically constant during network operation.

- **Collective event metrics** - its calculation involves the use of values from more than one node, considering the same event (events must be identifiable).
- **Collective network metrics** - its calculation involves the use of values from more than one node, considering a period of time.

The collective metrics value can be calculated in the path from the node to the sink, or just in the sink. In the first case, each node only has to report its performance data in one specific packet for each hop, each level passing the global collective view of its sub-network, lowering the bandwidth used and saving energy. Using the same packet loss example, each level should pass to the upstream levels all the packet loss of his entire sub-network for a specified period. In the second case, calculation of the collective metrics in the sink, the gain when compared to individual metrics is not on reducing the overhead, but on providing a global QoSensing perception to the network administrator and on enabling the set of specific alerts. In this case, collective metrics are only used to summarize the vast information gathered from the network. Collective metrics can always be used to enhance the network manager perception of the existing QoSensing, independently of the way the metrics reach the sink.

Both individual and collective metrics can be composed. Composed metrics are metrics built from different occurrences of the base metrics, which are composed in some manner (e.g. using aggregation functions such as average), usually in space or in time [54]. The

term composition was used originally in RFC2330 [6] and is used here in the sense of a mechanism to combine simple metrics into more complicated ones, where the values do not come from a single instances, but from the combination of multiple single instances. Composition in time provides the analysis of different time frames, to clear different behaviours of the network along the time (as an example, an average of the packet delivery delay from a node to the sink can be calculated every 30s). Composition in space provides for the division of the same metric along its network path (as an example, the sum of all delay hops to achieve the end-to-end-delay). If all metric occurrences concern the same individual node, composed individual metrics are obtained. If the occurrences include metrics from different nodes, with more than one occurrence of the metric per node considered, or collective metrics, composed collective metrics are obtained. In the latter case, two composition levels are obtained since the first is implicit to the notion of collective values, as information from different nodes is brought together. Composed individual metrics are built using consecutive values from the same node and consequently reduce the frequency of messages sent to the sink and increase the amount of time to get information from a specific node. Composed collective metrics contain multiple samples of values from multiple nodes, further lowering the overall number of messages in the network, as individual and composed individual values are no longer forwarded to the sink. As in the case of composed individual metrics, they also increase the amount of time necessary for the sink to get information from the network, as the update frequency is reduced. Fig. 4-2 depicts the different metrics presented and shows how they are calculated - the composition operation is represented by ◊.

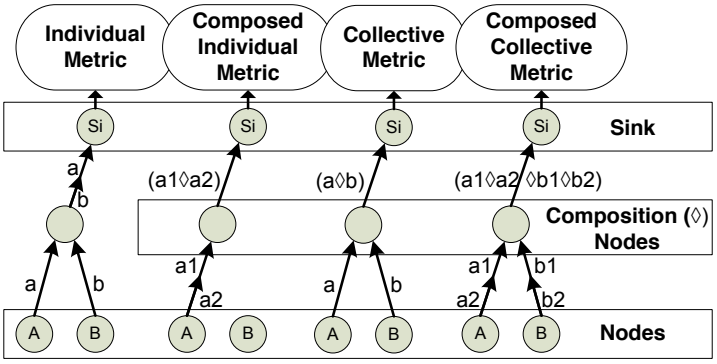


Fig. 4-2 - Individual, collective and composed metrics.

Next table summarizes the expected behaviour of each metric type described in a WSN scenario.

TABLE 4-3 – METRIC TYPES COMPARED

	Energy spent	Computation in nodes	Memory in nodes	Bandwidth required	Update rate possible	Evaluate global QoSensing
Individual	High	Low	Low	High	+	-
Collective in the sink	High	Low	Low	High	+	+
Collective in the network	Low	Med	Med	Low	+	+
Composed individual	Med	Med	Med	Med	-	+/-
Composed collective	Very Low	Med	Med	Very Low	-	+/-

In spite of being useful in hiding small inconsistencies of values, and in saving valuable resources, collective metrics may, under specific circumstances, hide a problem in the network, especially if its many nodes statistically prevent the error to be highlighted. To prevent these situations, a threshold may always be defined and if the value of each metric, as calculated in the node and/or transmitted by the node, surpasses that predefined limit, a special message can be sent to the sink, what will trigger a debugging process in the network.

In conclusion, due to the specific nature of WSNs, the use of collective metrics can be very useful as they:

- Hide the normal fluctuations of data that happen in most of WSN deployment environments;
- Demand less transmissions of data and save energy when calculated along the network;
- Give a context to the individual node performance values.

4.4.2 Special case of aggregation functions

Aggregation has always played an important role in WSNs, as its procedures are simple to understand, to implement and normally only demand simple calculations. Aggregation uses summarization functions such as the used in database query languages (e.g. *max*, *min*, *average*, *count*, *sum*, *median*). The simplicity of aggregation functions together with their ability to reduce the amount of data and to minimize redundancy, while improving data consistency and accuracy, make them a primary source for data composition in WSNs and explain its use in many WSN data fusion mechanisms [86]. Moreover, the delay introduced by the aggregation process is small, enabling real-time data delivery. Studies made in [60] reveal that significant energy gains are possible by using data aggregation, specially if the number of sources is large and close together, far from the sink.

In this chapter, aggregation functions will be used to compose individual and collective metrics.

Insecure aggregation functions

Some aggregation functions have been found to be insecure in what respects the way in which a reduced number of values can affect global aggregates. In [61] resilient aggregation, in a network security perspective, is addressed. Some popular aggregation functions were analysed to find out that many are unsuitable to cope with an attack to a single node, as the global function result was severely affected. Functions such as *average*, *sum*, *minimum* or *maximum* were found to be potentially insecure. If an attacker could take possession of some nodes, affecting their readings, the result of its action could potentially alter the computed aggregate significantly. A better option was to use the *median*, since even a large number of compromised nodes did not affect significantly the result. The function *count* was found to be secure as an attack on a node could not add more than 0 or 1 to the total. Truncation of values was also analysed, since the definition of a permissible value interval could help containing the influence of attackers. However it can also limit the dynamic range of values that can be obtained. According to the authors of the paper, a better approach is trimming, where a fixed percentage, of the higher and lower values, is dismissed (e.g. 5%), and where the aggregates are calculated over the remaining values.

While this work [61] focused network security problems, in the form of intentional attacks on some of the network nodes, its results can be easily extrapolated to a more general scenario. In the specific security scenario addressed, an attacker intentionally changed sensor values in order to affect the aggregated values. In a more general scenario, such as performance monitoring, out-of-bound performance values received from specific nodes may affect the overall measurement of performance. These out-of-bound values may be caused by different unintentional reasons, and are normally impossible to prevent. The ultimate objective is that every intentional or not intentional modification that affects a small number of nodes must not be translated into a global measurement that would be abnormally biased.

4.4.3 Approach to WSNs metrics definition

Considering the specificities of WSNs, the metrics to be used have some defining characteristics that influence the way they are used and calculated. When making the transition from the taxonomic tree of requirements to a valuable set of metrics, these characteristics must be taken into account. Next, some considerations and a proposed approach to the creation of metrics from the taxonomic tree of requirements introduced before, are presented.

Source of performance data

The performance parameters to be measured from a WSN can be either individual or collective (Fig. 4-3). Individual parameters relate to parameters that come from only one

node (even if subject to any type of composition), while collective parameters are a new type of parameter that results from the fact that its calculation involves the use of values from more than one node.

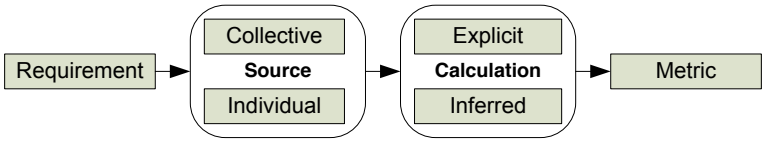


Fig. 4-3 - Obtaining metrics from requirements

Metrics calculation

Metrics can be calculated using two different approaches – active or passive. Data received in the sink node may contain metrics directly obtained from nodes (active approach) or metrics can be calculated in the sink node indirectly (passive approach). To distinguish these two scenarios, metrics can be divided in explicit and inferred (Fig. 4-3). Explicit metrics are obtained directly from the nodes (e.g. energy level) and sent to the sink/gateway node that may further treat them. Inferred metrics are obtained at the sink/gateway node indirectly, by analysing or making calculation on other data received (e.g. live nodes may be detected on receiving data collected by those nodes).

Metrics may be further composed (usually by using aggregation functions) in space and time, following the same principles defined previously for IPPM metrics (Section 4.3.4).

Whereas different requirements involve different metrics, different metrics may use the same parameters. As an example, delay can be used to calculate network latency or be part of a security metric, for example regarding intrusion detection. When metrics are not calculated at the nodes, reusing parameter values when possible ensures that minimum bandwidth is used by performance control data. In all cases, when calculating the metric itself, mainly when in the presence of collective parameters, one must have present that in general the cost of communication is far greater than the cost to execute several instructions, resolving the trade-off in favour of computation [28], and highlighting the advantages of using composed metrics.

WSN monitoring life cycle and metrics

Overall, the use of a complete framework for monitoring WSN performance can be divided in three distinct phases: *deployment*, *operation* and *debug&recovery* (Fig. 4-4).

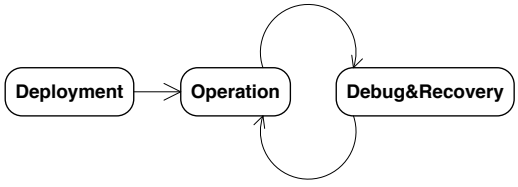


Fig. 4-4 – WSN monitoring life cycle

These phases correspond to the different life cycle phases of a network for which some metrics are more adequate than others. In Deployment, the framework of metrics will aid the network designer to establish the target performance before building the network, by having goals for each one of the relevant metrics, providing the guidelines for the specification and planning of the network. Also, during Deployment, the performance targets defined in the network project will be compared against the initial network tests, certifying that the initial performance goals are being met. If this is not the case, corrections should be made and network tests repeated, before the network starts its normal operation. At this phase all metrics can be used to fully test the network, with no restrictions. After deployment, during operation, the continuous monitoring of the network will assure that the required performance is under control and fulfils the initial expectations. Operation metrics are the minimum set of metrics that are required to be measured in order to detect malfunctions in the network and to assure that the initial requirements are being met. Using more metrics than necessary, or recurring to complex ones, will cause an increase of the overhead both in the nodes and in the network. The last phase, Debug&Recovery, occurs when a malfunction is detected. It includes all the additional metrics that can be used in the debug and recovery of the network, specially targeting specific nodes.

Concerns when creating new metrics to be calculated in WSNs nodes

When defining a set of metrics to be used in WSNs nodes, the restrictions of these specific networks must be taken into account (complex metrics calculated in the base station are out of the scope of our analysis). Considering the limited resources in terms of energy, memory and computation power, some characteristics were found to be of crucial importance:

- The metrics to be used should avoid unnecessary computation or wireless transmission at nodes, to save energy;
- The metrics should use the minimum memory possible – historic data should be avoided as possible in nodes;
- Metrics should be as generic and application independent, as possible;
- Real-time monitoring of the network should be possible.

Taking into account the referred characteristics, priority should be given to the inference of as many metrics as possible from the existing traffic, avoiding the transmission of unnecessary performance control data (lowering the overhead in the network and saving energy). Heavy calculations should be left to the sink (which has more computational power and fewer energy restrictions). In nodes, simple and easy to calculate metrics are advisable, and should provide for a real-time calculation of the performance of the network.

Despite being possible to use any metric in any phase (Deployment, Operation and Debug&Recovery), metrics should target and be designed for a specific phase in order to comply with its specific needs and avoid waste of resources. Specifically, operation phase should only rely on lighter and more essential metrics, to minimize the impact on a working network.

WSN and collective metrics

The use of collective values in WSN seems a natural approach when analysing their usual deployments and applications. In fact, to achieve reliability, typical WSN deployments many times use redundant nodes that send the same type of information. These nodes send similar sensed values, neither essential nor relevant, which normally vary slightly around a mean. Individual values are also subject to fluctuations due to the nature of WSNs (e.g. interferences, momentary losses of signal), diminishing the value of the information obtained from each value transmitted. Collective measurements can be an interesting option when analysing networks with redundant or very similar readings of the environment, or when the goal is to understand the global behaviour of the network. Furthermore, by composing data less transmissions are needed and less energy is spent. However, these new metrics cannot substitute the individual measurements in cases where each sensor has a specific sensing role.

In what respects performance metrics, collective metrics can also be of use to avoid the overhead that a continuous transmission of individual metrics would pose in the network, while maintaining a constant monitoring on the global performance. Further details and a specific evaluation of the different types of metrics are presented in next sections.

4.5 METRICS FOR THE NETWORK PERFORMANCE BRANCH OF THE TAXONOMY

In this section, a general set of metrics for the measurement of the requirements of the Network performance branch of the taxonomy will be proposed.

4.5.1 Objectives

From the requirements found necessary to characterize the QoSensing of a WSN with controlled performance, which resulted in the proposal of a new taxonomy, the Network performance branch was chosen as the basis for our metrics proposal. The taxonomy does not, by itself, provide the means to measure any of the requirement taxa. To be able to evaluate each requirement metrics are needed. The reason for selecting the Network performance branch relies on the objective to propose a reduced set of metrics that can be used in a generic industrial scenario. Based on the studies done for GINSENG, a complex industrial scenario, it was found that most of its priority requirements (see Section 3.6) were included in the Network performance branch of the proposed taxonomy (security

and mobility were the only exceptions but were not found as essential as the other requirements). In this section, a general set of metrics for measuring the requirements of the Network performance branch of the taxonomy will be proposed.

The Network group of our proposed taxonomy, specifically in its Performance branch, includes all the necessary requirements that influence the speed, quality and efficiency of the packets travelling in the network. This branch will be characterized and a set of metrics for its evaluation will be proposed. Details of this branch of the taxonomic tree are depicted in Fig. 4-5.

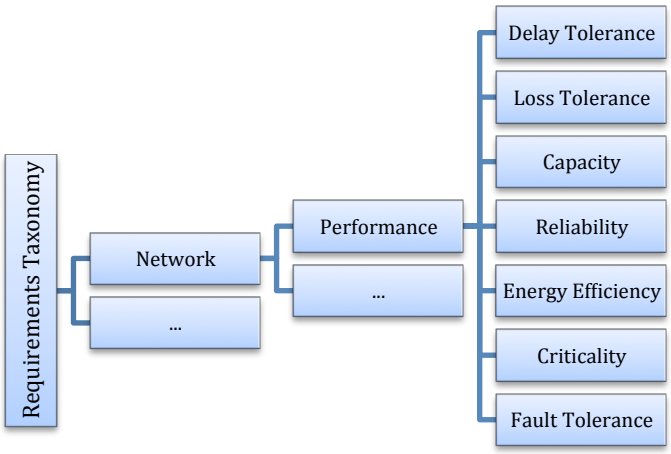


Fig. 4-5 – Performance branch of the requirements taxonomy presented in Section 3.4.2.

In our proposed taxonomy, seven groups of requirements are specified for the Network performance branch:

1. **Delay Tolerance** – specifies time bounds for the delivery delay of packets in the network;
2. **Loss Tolerance** – specifies loss bounds for data delivery in the network;
3. **Capacity** – measures the overall capacity of the network to the transmission of data;
4. **Reliability** – minimum assurances by the network that the sent packets reach destination without errors;
5. **Energy Efficiency** – specifies the amount of work per energy spent;
6. **Criticality** – specifies how the network deals with traffic priorities;
7. **Fault Tolerance** – specifies the tolerance of the network to permanent or temporary nodes failure.

It is not the aim of this section to present all the available and identified WSN performance metrics that can apply to the requirements of the taxonomy. Instead, the focus will be to propose the best generic, application independent metrics that were found necessary to characterize the network in order to respond to the taxonomy tree of requirements. The metrics chosen must respect the specificities of WSNs and follow the

proposed approach as discussed in Section 4.4.3. These metrics will be later categorized in three groups according to the situation where they are mostly used: Deployment, Operation and Debug & Recovery. The information necessary for their calculation, both from the *Management Information Base* (MIB) of the node as from the packet header, is also presented. MIB in this context is used generically as a management table not directly correlated to the MIB used by the *Simple Network Management Protocol* (SNMP).

To use collective event metrics, events have to be identifiable individually and/or by event type, and this information must be sent in the data packets.

4.5.2 Definition of metrics

To enable the measurement of each of the requirements addressed in the Network performance branch of the taxonomy, a set of metrics will now be proposed. The method of metrics composition chosen relies on aggregation functions.

Next, each metric of the framework will be specified.

Being *N* the set of nodes in the network, $P_{n1 \rightarrow n2}$ all packets sent from node *n1* to *n2,* $pkt_{n1 \rightarrow n2}$ the number of packets sent from node *n1* to *n2* ($\#P_{n1 \rightarrow n2}$), $P_{n1 \leftarrow n2}$ all packets received in node *n1* from *n2,* $pkt_{n1 \leftarrow n2}$ the $\#P_{n1 \leftarrow n2}$, *t* the time when the packets are sent, *[ti,tf]* the time interval and $En_{n1}(t)$ the energy of node *n1* at time *t*, the proposed metrics will be defined as:

1) Delay tolerance

The delay in a packet switching network expresses the time a packet spends travelling from the original sender to its destination, including propagation and transmission time. A packet delay is measured by subtracting the destination timestamp from the original sent timestamp. In the case of packets sent from nodes to sink or from sink to node (i.e. end-to-end) the term 'delivery delay' will be used. The clocks between sender and destination must be synchronized in most cases. The time unit used is the second (the same specified by IPPM [6]).

Primary metrics:

- One-way node delivery delay (Type: individual; Place of calculation: sink)
  - Time spent by the packet from the original sender (a) node to the sink (end-to-end). Will be defined as:

$$\boldsymbol{dd_{a \rightarrow sink}(i)}, \text{ a} \in \text{N, i} \in \boldsymbol{P_{a \rightarrow sink}}. \quad (4\text{-}1)$$

- One-way sink delivery delay (Type: individual; Place of calculation: node)
  - Time spent by a packet from the sink to a node. The calculated value must be sent to the sink in other packet. Will be defined as:

$$\boldsymbol{dd_{sink \rightarrow a}(i)}, \text{ a} \in \text{N, i} \in \boldsymbol{P_{sink \rightarrow a}}. \quad (4\text{-}2)$$

- Query delivery delay (Type: individual; Place of calculation: sink)
  - Time spent by a packet from the sink to a node together with the response from the node to the sink. Does not need clock synchronization.
- Actuation delivery delay (Type: individual; Place of calculation: node)
  - Time spent in sending a packet from a node to the sink, followed by the response of the sink to the node. The calculated value must be sent to the sink in other packet. Does not need clock synchronization.
- Node delivery delay variation (Type: individual; Place of calculation: sink)
  - Difference between the delays of two consecutive packets arriving from the same node to the sink (may be positive, 0 or negative).
- Delay per hop (Type: individual; Place of calculation: sink or intermediate nodes)
  - Individual delays of packets measured between single hops. Reports must be periodically sent to the sink with this information. To avoid a high number of packets and to minimize the memory wasted, average values should be recorded and then periodically sent to the sink.
- Max/Min/Average/StDev node delivery delay (Type: aggregated; Place of calculation: sink)
  - Maximum, Minimum, Average and Standard Deviation of delay from packets sent by a specific node (a) to the sink in a specified time interval. It is calculated in the sink. Will be defined as:

$$MaxNodeDD_{a\text{-}sink}=max(\{dd_{a\rightarrow sink}(i)\}_{i=1,\ldots,pkt_{a\rightarrow sink}}),$$

$$MinNodeDD_{a\text{-}sink}=min(\{dd_{a\rightarrow sink}(i)\}_{i=1,\ldots,pkt_{a\rightarrow sink}}),$$

$$AvgNodeDD_{a\text{-}sink}=avg(\{dd_{a\rightarrow sink}(i)\}_{i=1,\ldots,pkt_{a\rightarrow sink}}),$$

$$StDevNodeDD_{a\text{-}sink}=stdev(\{dd_{a\rightarrow sink}(i)\}_{i=1,\ldots,pkt_{a\rightarrow sink}}),$$

$$a \in N,\ i \in \boldsymbol{P_{a\rightarrow sink}},\ \boldsymbol{t_i \leq T(i) \leq t_f} \quad (4\text{-}3)$$

- Collective event delivery delay (Type: collective; Place of calculation: sink or intermediate nodes)
  - Difference between the time the data was first obtained in any node and the time when the last packet concerning the event, from all targeted nodes, arrived to the sink.

- Max/Min/Average/StDev collective network delivery delay (Type: collective; Place of calculation: sink)
  - Maximum, Minimum, Average and Standard Deviation of delay considering packets received in sink (or calculated in intermediate nodes), from all nodes, in a period of time. It is calculated in the sink. Will be defined as:

$$MaxCollectiveNetworkDelay = max(\{dd_{x\rightarrow sink}(i)\}_{x=1,\dots,n;i=1,\dots,pkt_{x\rightarrow sink}}),$$

$$MinCollectiveNetworkDelay = min(\{dd_{x\rightarrow sink}(i)\}_{x=1,\dots,n;i=1,\dots,pkt_{x\rightarrow sink}}),$$

$$AvgCollectiveNetworkDelay = avg(\{dd_{x\rightarrow sink}(i)\}_{x=1,\dots,n;i=1,\dots,pkt_{x\rightarrow sink}}),$$

$$StDevCollectiveNetworkDelay = stdev(\{dd_{x\rightarrow sink}(i)\}_{x=1,\dots,n;i=1,\dots,pkt_{x\rightarrow sink}}),$$

$$x \in N,\ i \in \boldsymbol{P_{x\rightarrow sink}},\ n=\#N,\ \boldsymbol{t_i \leq T(i) \leq t_f} \quad (4\text{-}4)$$

Additional metrics:

- Max/Min/Average/StDev delivery delay variation (Type: aggregated; Place of calculation: sink)
  - Maximum, Minimum, Average and Standard Deviation of delivery delay variation from packets sent by a specific node to the sink in a specified time interval.
- Max/Min/Average/StDev sink delivery delay (Type: aggregated; Place of calculation: node)
  - Maximum, Minimum, Average and Standard Deviation of delay from packets sent from sink to a specific node in a period of time. The individual values have been sent to the sink where the calculation takes place.
- Max/Min/Average/StDev collective event delivery delay (Type: collective; Place of calculation: sink):
  - Maximum, Minimum, Average and Standard Deviation of collective delays, concerning multiple events, in a period of time. It is calculated in the sink.
- Max/Min/Average/StDev collective delay per hop (Type: collective; Place of calculation: sink or intermediate nodes):
  - Maximum, Minimum, Average and Standard Deviation of collective delays per hop from packets sent from all or part of the network nodes to sink, in a period of time.

2) Loss tolerance

In a packet switching network the loss expresses the number of packets lost (or arriving after a predefined time) during the communication of two hosts in a specified time interval. The packet loss is measured by subtracting the number of packets sent to the number of packets that arrive destination in a specified time interval. To be able to

calculate the lost at the destination node the packets must have sequential identifiers or the source node must periodically send a packet specifying the number of packets sent to date. The unit used is the number of packets lost during the period, as measured in sink.

Primary metrics:

- One-way node loss (Type: individual; Place of calculation: sink):
  - Number of lost packets from the original sender node (a) to the sink in a specified time interval (end-to-end). Will be defined as:

$$\boldsymbol{loss_{a\rightarrow sink}(t_i, t_f)},\ a \in N. \tag{4-5}$$

- One-way sink loss (Type: individual; Place of calculation: node):
  - Number of lost packets from the sink to a node (a) in a specified time interval. The value must then be sent to the sink in other packet. Will be defined as:

$$\boldsymbol{loss_{sink\rightarrow a}(t_i, t_f)},\ a \in N. \tag{4-6}$$

- Collective event loss (Type: collective; Place of calculation: sink with or without collaboration from intermediate nodes):
  - Total number of lost packets considering all the packets sent by all source nodes, and related to the same event (Event), to the sink, in a specified time interval. Will be defined as:

$$CollectiveNetworkLoss = \sum_{x=1}^{n} \boldsymbol{loss_{x\rightarrow sink}}\ (\boldsymbol{t_i, t_f}),\ x \in Event,\ n=\#N \tag{4-7}$$

- Collective network loss (Type: collective; Place of calculation: sink):
  - Total number of lost packets considering all the packets sent by all the source nodes to the sink, in a specified time interval. Will be defined as:

$$CollectiveNetworkLoss = \sum_{x=1}^{n} \boldsymbol{loss_{x\rightarrow sink}}\ (\boldsymbol{t_i, t_f}),\ x \in N,\ n=\#N \tag{4-8}$$

Additional metrics:

- Loss per hop (Type: aggregated; Place of calculation: sink with or without collaboration from intermediate nodes):
  - Loss of packets measured between single hops. Reports are sent periodically sent to the sink with this information. To avoid a high number of packets and to minimize the memory wasted, total values should be recorded in nodes and then periodically sent to the sink.
- Loss length per node (Type: individual; Place of calculation: sink or node):
  - Counts the number of consecutive lost packets from a specific node, and is an indicator of the burstiness.

- Loss distance per node (Type: individual; Place of calculation: sink or node):
  - Counts the number of packets between two lost packets sent by the same node. It indicates the frequency of the loss.
- Total retransmissions by node (Type: aggregated; Place of calculation: node):
  - Sum of total retransmissions made by a node in a period of time.
- Max/Min/Average/StDev node loss (Type: aggregated; Place of calculation: sink or node):
  - Maximum, Minimum, Average and Standard Deviation of specific node losses when sending packets to other node, considering different time periods.

3) Capacity

Measuring the capacity of a link is no easy task to do or even to define, while measuring the capacity of a network is even more difficult. Capacity varies with the protocol layer, with type of packets, with the conditions of the link. For the purpose of the WSN communication performance metrics capacity is going to be defined as the maximum sustainable throughput of L2 unique data (excluding retransmissions) that is received by the sink, and is measured in bytes/sec. The available capacity end-to-end will be the minimum capacity of each segment of the network. It is measured in bytes per second or, in some cases, as a percentage.

Primary metrics:

- Capacity per node (Type: individual; Place of calculation: sink):
  - Maximum amount of sustainable throughput (bytes/sec), excluding retransmissions, of a node. It is calculated by counting the number of bytes, in unique L2 packets, received from a specific node, by the sink.
- Capacity use per node (Type: individual; Place of calculation: sink):
  - Percentage of capacity of a node that is being used - is obtained by dividing the current node throughput by its previously measured capacity.
- Collective network capacity (Type: collective; Place of calculation: sink):
  - Maximum amount of sustainable throughput (bytes/sec), excluding retransmissions. It is calculated by calculating the total number of bytes received from the network, in unique L2 packets, by the sink.
- Collective network capacity use (Type: collective; Place of calculation: sink):
  - Percentage of capacity that is being used - is obtained by dividing the current throughput by the previously measured capacity.

4) Network reliability

The reliability of a network expresses its capacity of delivering packets to destination without errors and in a previously defined timeframe. In order to assess for the reliability, sent and received packets must be counted. Reliability will be presented as a percentage

of delivered packets. If a sequence number is assumed, the receiving node has to track the received packets and to detect the missing packets. If the sequence numbers do not exist, periodically each node must report the number of packets sent. When considering end-to-end transmissions the term 'delivery reliability' will be used. To calculate the reliability of a WSN the proposed metrics derive from the Packet Delivery Ratio, measured as follows:

$$\boldsymbol{Packet\ Delivery\ Ratio}(\%) = \frac{\boldsymbol{Packets\ received}}{\boldsymbol{Packets\ sent}} \times \mathbf{100} \tag{4-9}$$

This metric is evaluated with a number between 0 and 100, corresponding to the percentage of delivered packets.

Primary metrics:

- Node delivery reliability (Type: individual; Place of calculation: sink):
  - Reliability of the connection from a specific node (a) to the sink, considering packets it sends and those received from sink in a time period. It corresponds to the Packet Delivery Ratio:

$$PktDelivRatio_{a \rightarrow sink}\ (\%) = (\boldsymbol{pkt_{sink \leftarrow a}}) / (\boldsymbol{pkt_{a \rightarrow sink}}) \times 100,\ a \in N \tag{4-10}$$

- Sink delivery reliability (Type: individual; Place of calculation: sink):
  - Reliability of the connection from the sink to a node (a), considering the packets it sends and receives from the node in a period of time. The measurement must then be sent to the sink in a packet.

$$PktDelivRatio_{sink \rightarrow a}\ (\%) = (\boldsymbol{pkt_{a \leftarrow sink}}) / (\boldsymbol{pkt_{sink \rightarrow a}}) \times 100,\ a \in N \tag{4-11}$$

- Collective event reliability (Type: collective; Place of calculation: sink):
  - Reliability of all nodes of the network that participate in the same event (Event). It is calculated using the sum of all packets received by the sink and those sent by nodes, relating to a specific event.

$$CollectivePktDelivRatio\ (\%) = (\sum_{x=1}^{n} \boldsymbol{pkt_{sink \leftarrow x}}) / (\sum_{x=1}^{n} \boldsymbol{pkt_{x \rightarrow sink}}) * 100, \quad x \in Event,\ n = \#N \tag{4-12}$$

- Collective network reliability (Type: collective; Place of calculation: sink):
  - Reliability of all nodes of the network when considered together. It is calculated using the sum of all packets received by the sink and those that were sent in a period of time.

$$CollectivePktDelivRatio\ (\%) = (\sum_{x=1}^{n} \boldsymbol{pkt_{sink \leftarrow x}}) / (\sum_{x=1}^{n} \boldsymbol{pkt_{x \rightarrow sink}}) * 100, \quad x \in N,\ n = \#N \tag{4-13}$$

Additional metrics:

- Total packets sent by node (Type: individual; Place of calculation: node):
  - Measures the total number of packets sent by the node in a period of time.
- Total packets received by node (Type: individual; Place of calculation: node):
  - Measures the total number of packets received by the node in a period of time.
- Total packets received by the sink (Type: individual; Place of calculation: node):
  - Measures the total number of packets received by the node in a period of time.
- RSSI average per node (Type: individual; Place of calculation: node):
  - RSSI of the packets received from a specific node measured between single hops. Reports must be periodically sent to the sink by the receiving node, with this information. To avoid a high number of packets and to minimize the memory wasted, average values should be recorded and then sent to the sink periodically.
- Reliability per hop (Type: individual; Place of calculation: sink or intermediate node):
  - Reliability of the connection between nodes measured between single hops. Reports must be periodically sent to the sink with this information. To avoid a high number of packets and to minimize the memory wasted, average values should be recorded and then periodically sent to the sink.
- Average/StDev node delivery reliability (Type: aggregated; Place of calculation: sink):
  - Measures the Average/Standard Deviation of the reliability of a node considering different periods.
- Average/StDev sink delivery reliability (Type: aggregated; Place of calculation: sink):
  - Measures the Average/Standard Deviation of the reliability of the transmission from the sink to a node, considering different periods.

5) Energy efficiency

Measuring the energy in WSN is crucial as most of nodes run on batteries. Also, it is important to know how efficient nodes are when using the available energy. The energy is measured in Volts, Watts or Joules and the energy efficiency is the amount of work done (number of operations) per Watt.

Primary metrics:

- Energy level per node (Type: individual; Place of calculation: node):
  - Total energy that is available in the node.

- Total energy spent per node (Type: individual; Place of calculation: node)
  - Maintains a record of all the energy spent by a node a, measured since a specific point in time (ti) or since it started operating.

$$\Delta En_a(t,ti)=En_a(t)-En_a(ti) \quad (4\text{-}14)$$

- Average energy spent per message sent by a node (Type: aggregated; Place of calculation: node):
  - Total energy a node spends in its operation divided by the number of messages it sends in the same period.

$$AvgEnergySpentPerMessage = \Delta En_a / (\sum_{x=1}^{n} \boldsymbol{pkt_{a\rightarrow x}}),\ a, x \in N,\ n=\#N \quad (4\text{-}15)$$

- Collective energy spent in the network/group of nodes (Type: collective; Place of calculation: sink):
  - Maintains a record of all the energy spent by the nodes in network, measured since a specific point in time (ti) or since it started operating.

$$TotalEnergyWaste = \sum_{x=1}^{n} \Delta \boldsymbol{En_x(t, ti)},\ x \in N,\ n=\#N \quad (4\text{-}16)$$

- Collective average energy spent per message sent in the network (Type: collective; Place of calculation: sink):
  - Total energy spent in the network divided by the number of messages it produces and sends.

$$AvgEnergySpentPerMessage = \sum_{x=1}^{n} \Delta \boldsymbol{En_x(t, ti)} / (\sum_{x=1}^{n}\sum_{y=1}^{n} \boldsymbol{pkt_{x\rightarrow y}}),\ x,y \in N,\ n=\#N \quad (4\text{-}17)$$

Additional metrics:

- Transmission time per node (Type: individual; Place of calculation: node):
  - Time, in seconds, spent in radio transmitting, measured since the beginning of its operation.
- Listen time per node (Type: individual; Place of calculation: node):
  - Time, in seconds, spent in radio listening, measured since it started operating.
- Activity-time per node (Type: individual; Place of calculation: node):
  - Time spent by the node in active mode, measured since it started operating.
- Idle-time per node (Type: individual; Place of calculation: node):
  - Time, in seconds, spent by the node in sleep mode, measured since it started operating.
- Duty-cycle per node (Type: individual; Place of calculation: node):
  - Measures the ratio between time spent by the node in active state by the total time of operation.

6) Traffic criticality

In a network where different traffic may be present, where the resources are scarce and where urgent messages may have to be delivered with strict time bounds, it is necessary to measure how critical traffic is supported. These critical packets may arise, for example, from predefined triggers that measure values in nodes or from messages from the sink to actuators. To measure how critical traffic is supported in the network, its performance will be compared to normal traffic. It is not needed to know, for the purpose of performance evaluation, the exact mechanisms used to raise the priority of critical messages. It is just necessary to assure that they work.

Primary metrics:

- Critical packets node delivery delay (Type: individual; Place of calculation: sink):
  - Measures the delay, in seconds, of a critical message from a node to the sink.
- Critical packets sink delivery delay (Type: individual; Place of calculation: node):
  - Measures the delay, in seconds, of a critical message from the sink to the destination node.

Additional metrics:

- Average speedup of critical messages by node (Type: individual; Place of calculation: sink):
  - Measures how faster, in average, critical packets are when compared to normal packets sent by the same node. It is measured by dividing the average delay per node of normal packets by the average delay per node of critical packets.
- Critical packet compliance (Type: aggregated; Place of calculation: sink):
  - Measures the % of critical packets that have a delay below a specific threshold set in seconds.
- Average critical packet node delivery delay (Type: aggregated; Place of calculation: sink):
  - Average time, in seconds, spent by a critical packet from the original sender node to the sink (end-to-end).
- Average critical packet sink delivery delay (Type: aggregated; Place of calculation: node):
  - Average time, in seconds, spent by a critical packet sent from the sink to its destination node. The calculated value must then be sent to the sink in other packet.

7) Fault tolerance

Fault tolerance in a WSN is the ability of the network to tolerate faults that lead to service failures. It is an aspect of the resilience of the network. Minimizing the detection time of a node failure reduces the time during which other nodes are forwarding packets to dead

neighbours, allows the network to try to recover faster and also minimizes routing problems. On minimizing this time, care must be taken not to increase the false positives.

Proposed metrics:

- Number of active nodes (Type: aggregated; Place of calculation: sink):
  - Gives the number of active nodes in the network. If all nodes are active there is no fault. An adjustable time limit must be defined in order to distinguish between faults and temporary faults (automatically adapted to the maximum times of disruption of each node). Being $T_{LastContact}(x)$ the time when a last contact was received from node x and ALIVE_THRESHOLD a value defined as a time threshold above which the node is considered dead or malfunctioning, the number of active nodes is calculated as:

$$Alive(x) = \begin{cases} 1, T_{LastContact}(x) \leq ALIVE\_THRESHOLD, x \in N \\ 0, T_{LastContact}(x) > ALIVE\_THRESHOLD, x \in N \end{cases}$$

$$ActiveNodes = \sum_{x=1}^{n} \boldsymbol{Alive(x)}, \boldsymbol{x} \in \boldsymbol{N},\ \boldsymbol{n} = \#\boldsymbol{N} \quad (4\text{-}18)$$

Additional metrics:

- Average time to detect node failure (Type: aggregated; Place of calculation: sink or intermediate node):
  - Average elapsed time between when a node was considered in failure and the time it sent the last packet. Measured in seconds.
- Average time to recover (Type: aggregated; Place of calculation: sink or intermediate node):
  - Average elapsed time between the time when a node was considered in failure and the time it sends a new packet. Measured in seconds.
- Average downtime per node (Type: aggregated; Place of calculation: sink or intermediate node):
  - Average time a node was in failure, measured in a period of time. Measured in seconds.

Metrics by network life phase

Although every metric can be used at any time, some specific metrics target a specific phase in the life of the network. In Table 4-4 the performance metrics presented before are categorized by the phase in which they are most necessary (some may be used in different phases).

TABLE 4-4 – GENERAL FRAMEWORK OF NETWORK PERFORMANCE METRICS BY PHASE

	Deployment In project time and deployment	**Operation** Minimal metrics to assure network performance	**Debug & Recovery**
Delay Tolerance	▪ Average node delivery delay; ▪ Average sink delivery delay;	▪ One-way node delivery delay; ▪ One-way sink delivery delay; ▪ Query delivery delay; ▪ Actuation delivery delay; ▪ Collective event delivery delay; ▪ Avg. node delivery delay variation; ▪ Avg. Collective delay per hop	▪ Delay per hop; ▪ Node delivery delay variation; ▪ Average collective event delivery delay; ▪ Average collective network delivery delay;
Loss Tolerance	▪ Average node loss;	▪ One-way node loss; ▪ One-way sink loss; ▪ Collective event loss; ▪ Collective network loss;	▪ Loss per hop; ▪ Total retransmissions by node; ▪ Loss length per node; ▪ Loss distance per node;
Capacity	▪ Capacity per node; ▪ Collective network capacity;	▪ Collective network capacity use;	▪ Capacity use per node;
Reliability	▪ Average node delivery reliability; ▪ Average sink delivery reliability;	▪ Node delivery reliability; ▪ Sink delivery reliability; ▪ Collective event reliability; ▪ Collective network reliability; ▪ RSSI average per node;	▪ Reliability per hop; ▪ Total packets sent by node; ▪ Total packets received by node; ▪ Total packets received by the sink;
Energy Efficiency	▪ Average energy spent per message sent by a node; ▪ Energy level per node; ▪ Duty-cycle per node; ▪ Collective total energy spent in network; ▪ Collective average energy spent per message sent in network;	▪ Average energy spent per message sent by a node; ▪ Energy level per node; ▪ Total energy spent per node;	▪ Transmission time per node; ▪ Activity-time per node; ▪ Idle-time per node; ▪ Listen time per node; ▪ Duty cycle per node;
Criticality	▪ Average critical packet node delivery delay; ▪ Average critical packet sink delivery delay;	▪ Critical packets node delivery delay; ▪ Critical packets sink delivery delay;	▪ Average speedup of critical messages by node; ▪ Critical packet compliance;
Fault Tolerance	▪ Average time to detect node failure;	▪ Number of active nodes;	▪ Average time to recover; ▪ Average downtime per node;

Basic information needed for metric calculation

The metrics presented assume that a small amount of information is saved in the MIB of each node. As nodes have restrictions in the memory available, this information should not exceed the minimum necessary. Table 4-5 presents the necessary fields.

TABLE 4-5 – NECESSARY FIELDS FOR THE MIB OF THE NODE

▪ Total energy spent in the node; ▪ Energy remaining in the node; ▪ Total listening time; ▪ Total transmitting time; ▪ CPU time;	▪ Idle time; ▪ Uptime; ▪ Number of packets sent; ▪ Number of forwarded packets; ▪ Number of packets retransmissions;	▪ Number of received packets; ▪ *Average* RSSI from received packets; ▪ *Average* RSSI from received packets from a specific node; ▪ RSSI measured in the last packet received; ▪ Response time from sink - delay average;

In the first column, fields that are necessary to calculate the energy of the node by software are presented [87]. Second and third columns save operational data related to the node activity and of its interaction with neighbours.

Also, to calculate the metrics, some information contained in the packets being transmitted is needed. That information is listed in Table 4-6.

TABLE 4-6 – NECESSARY FIELDS IN DATA PACKET

▪ Source ID; ▪ Destination ID;	▪ Sender timestamp; ▪ Number of hops;	▪ Packet sequence number; ▪ Event type;

4.6 ANALYSIS OF DIFFERENT TYPES OF METRICS IN THE EVALUATION OF THE QOSENSING IN WSNS

One of the concerns of using performance metrics in WSNs is that their use contributes to the degradation of the performance that they want to measure. This is more obvious when using performance monitoring in operation phase. Usually, the feedback from the network is achieved by having a continuous flow of performance data from each node, which is forwarded hop by hop to the sink. As a consequence, the increase of the traffic associated with the performance measurement wastes bandwidth, creates interference, depletes energy from nodes and contributes to higher packet losses and higher delays. This is due to the high demands of data transmission in the sensor nodes and in all the other that must route the packets to the sink (in multi-hop networks). A smaller volume of performance data traffic may be achieved by increasing the time between each feedback report. However, at the same time, the control over the state of each node decreases and the time to detect that a node has failed increases. To reduce overhead, performance can also be deduced by inferring metrics from regular packets (e.g. delay – using send time information from packets, loss - by using sequence numbers in packets). The inclusion of some specific performance data in each data packet is also possible. However, the latter option is not always possible or needed as nodes may deliver data at a different rate of their performance control needs.

To evaluate the contribution, advantages and disadvantages of each type of metric - individual, collective and composed (composition will be achieved by using aggregation

individual metrics) - in the assessment of the QoSensing of a WSN, tests using simulation were made. The metrics chosen as an example are addressed next.

4.6.1 Metrics Selection

To compare and evaluate the use of different types of metrics in the global assessment of the QoSensing of a WSN with controlled performance and with real-time demands, the requirements found more important for GINSENG [21] (message delay, message delivery reliability, fault tolerance, energy consumption and energy efficiency) were used. To address each of the specific requirements, specific metrics were chosen and adapted from the framework available and presented before (Table 4-4).

Individual metrics

- One-way (individual) node delivery delay;
- One-way (individual) node loss;
- Node delivery reliability;

Aggregated metrics

- Max/Min/Average/StDev node delivery delay;
- Total energy spent per node;
- Average energy spent per message sent by a node.

Collective metrics

- Max/Min/Average/StDev collective network delay;
- Collective network loss;
- Collective network reliability;
- Total energy spent in the network/group of nodes;
- Average energy spent per message sent in the network;
- Number of active nodes;
- Average time to detect node failure.

The proposed metrics can also be calculated considering packets sent to other nodes besides the sink, as for a group of nodes instead of all nodes in the network. Time intervals when considering collective metrics depend on the specific requirements of the monitoring – larger intervals lower the network overhead but also decrease the real-time monitoring capabilities, while lower times increase network overhead but allow for a more detailed monitoring.

4.6.2 Collection methods used

In this analysis, 3 collection methods are used. Metrics are inferred in sink from arriving packets, individual performance packets from each node are directly sent to the sink and collective metrics are calculated along the path until reaching the sink.

Since the resources in nodes are limited, it would always be better to send individual performance data from each node directly to the sink, where a more sophisticated analysis could be processed. However, it would not only contribute heavily to the reduction of the lifetime of nodes, but also waste a valuable amount of the available bandwidth, lowering in the process the existing network performance.

In order to minimize the impact that a real-time performance monitoring system has in the WSN, the use of collective metrics with alerts is also analysed, together with the use of inferred metrics. Instead of forwarding the individual performance packets from downstream nodes in the delivering tree, intermediate nodes may forward collective metrics. As the collective metrics are calculated using the individual values received, they express a global view of the network below. However, if individual values are found to be out of the healthy range, a specific alert is generated, which is then immediately forwarded to the sink. As an example, the value of the energy of each specific node is not necessary, as long as it is above a specific value that can be pre-establish. In this case, the metric reaching the sink could be the total network nodes energy and, if any of the nodes is found to be below a specific threshold, an individual alert is sent. The use of alerts together with collective metrics enables that a control can be made at each hop of the network towards the sink and, at the same time, lowers the overhead associated with the performance collection process. The use of inferred data, as stated before, also helps to lower the number of performance packets (or their size) as some metrics are indirectly calculated from the packets that arrive at the sink.

A full protocol to collect metrics from the network will be presented in Chapter 5.

In the evaluation of metrics presented in the next section, all the collection methods referred were used.

4.6.3 Simulation and analysis

The scenario simulated will approach the topology used in the GINSENG test bed [88], which is focused in industrial environments, featuring a small number of nodes with previously studied positions.

4.6.3.1 Simulation Scenario

The first tests were made using a simulation consisting of a scenario with 13 nodes (0-12). Collective metrics with alerts and scalability tests were made with scenarios of 22 nodes (0-21) and 31 nodes (Fig. 4-6).

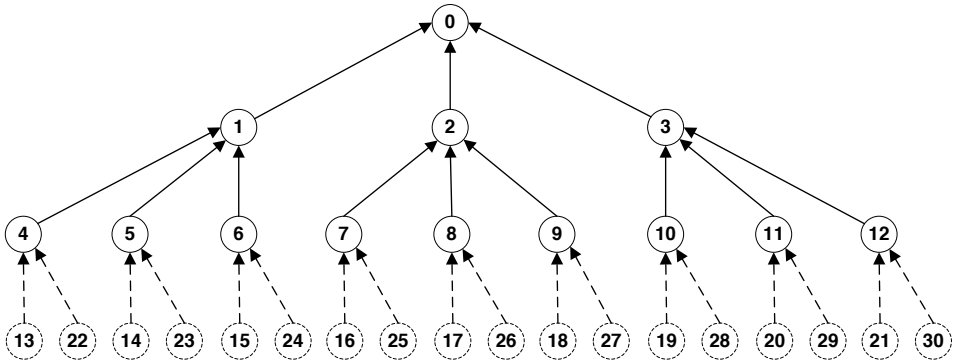


Fig. 4-6 – Simulation scenario nodes

Simulation was made using the Cooja simulator [89]. The area simulated was of 170x60m. Transmission range was set to 50m and interference range to 80m. All nodes report to the same sink (node 0) in a hierarchical way and each leaf node sends a packet with performance data (the number of data packets already sent and the node's energy) every 3 seconds. As other nodes just forward the data received, the congestion level is low. Send and receive ratio was set to 100% in the simulation setup file, which means that nodes will try to deliver 100% of the packets. Each leaf node sends a maximum of 100 packets after which the simulation finishes. To emulate some network problems, 3 packets sent by node 8 will be lost and node 10 will cease sending packets after packet number 3 (simulating a dead node). Additionally, from simulation time 20-30 s ($0.2x10^5$ – $0.3x10^5$ ms) and 150-250 s ($1.5x10^5$ – $2.5x10^5$ ms), two additional nodes were created to interfere with the communications in the WSN. This interference only affects the sub-tree of node 3. ContikiMAC Radio Duty Cycling Protocol [90] was used in all nodes, including the sink. The reason to include the protocol in the sink regarded its possible use as a wireless gateway to a central station. Energy spent was calculated by a software-based power profiling mechanism of Contiki [23]. The mechanism runs directly on the sensor nodes and provides real-time estimates of the current energy consumption [87].

In the first tests all metrics were calculated in the sink using either explicit and inferred performance data. The reason of making all the calculations at the sink is to focus the tests in the differences between each type of metric and on the information each can provide. Examples of explicit data are the node's energy values transmitted in each packet. Delay calculation is an example of inferred data obtained at the sink node. The second part of the simulation tests addresses the use of collective metrics together with alerts, in order to lower the impact of the performance monitoring.

4.6.3.2 Metrics Comparison and Analysis

Individual, aggregate and collective metrics were calculated and are depicted along with some analysis of their benefits and limitations.

Individual metrics

Fig. 4-7 and Fig. 4-8 show simulation results for the individual delay and loss of nodes 5 and 12 (node 12 was affected by several interference as stated before).

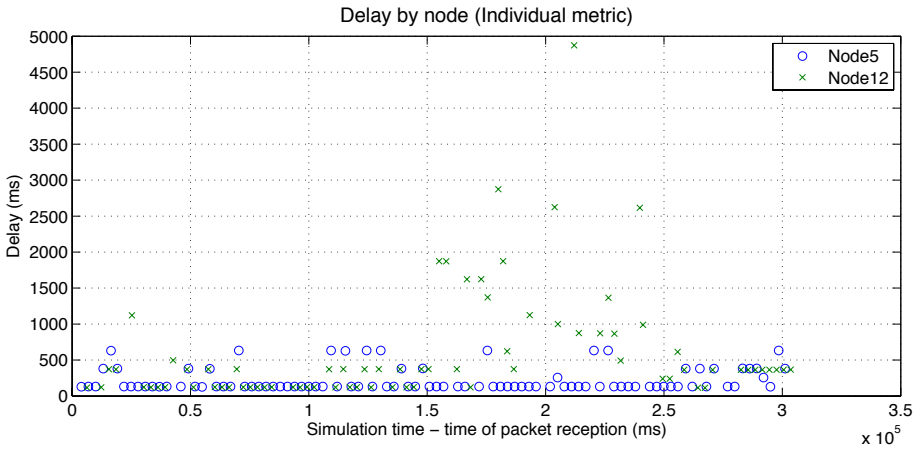


Fig. 4-7 - Delay in nodes 5 and 12.

Considering all the nodes from the network, the delay range was found to be from 64 to 4872 ms. This extreme variation is not unusual in a WSN if all packets are considered along operation time, especially if interference exists. A simple interference may lead to higher values, not being possible to assume a severe malfunction of a node from an isolated measurement. Small packet losses happened during the time where interference raised. The interference affected node 12 as can be seen in Fig. 4-8. In the period from 150-250 s the number of packets reported as lost became more frequent. Also, despite a huge amount of constant packet losses may be indicative of a problem, the same may or may not happen when dealing with small losses. To assess its importance one must know the behaviour of neighbour nodes to clarify the need of network debugging.

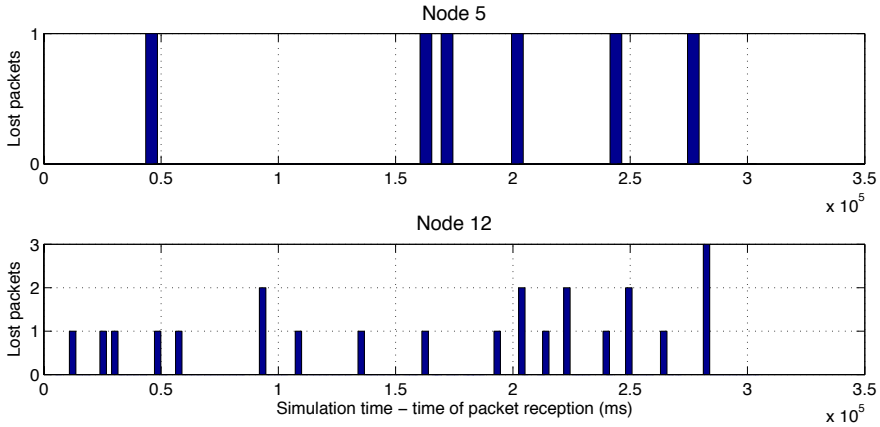


Fig. 4-8 - Losses in nodes 5 and 12.

Energy was also individually evaluated. Fig. 4-9 depicts the energy spent along the simulation time by nodes 4, 5, 11 and 12. Energy spent takes into consideration different transmission hazards that may lead to temporary difficulties in transmitting data. As nodes 4 and 5 did not suffer interferences, beside those inherent to the sending of the packets, their spent energy was proportional to the simulation time. Nodes 11 and 12 suffered interferences from external sources between simulation time 20-30 s and 150-250 s, what implied additional energy spent in those periods, especially from 150-250 s, where the energy spent increased significantly. Energy spent by Node 4 was also measured and ranged from 3 to a total of 621 mJ (respectively after sending the first packet and the last). The node with minimum spent energy was node 10, where packets ceased being transmitted after 4 packets sent. Node 11, the most affect node by the interference, spent a total of 759 mJ during all simulation time.

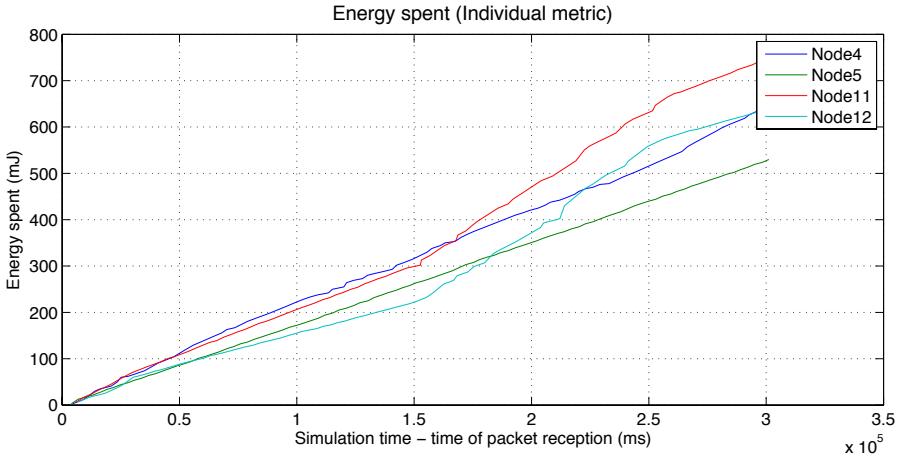


Fig. 4-9 – Total energy spent along simulation, for nodes 4, 5, 11 and 12.

Only using individual metrics analysis provides no clear assessment or conclusion on the performance of the all network. Even if some value ranges were previously established to bound acceptable individual metrics, only the evolution of these metrics along time, and the comparison to metrics from neighbour nodes, can provide for clear global performance knowledge in a WSN environment.

As for the detection of the network problems created, it is possible to detect the death of node 10 as it stopped transmitting and to count the lost packets, assuming that every data from every node is analysed. However, both problems, and also the interference created by exterior nodes, cannot be analysed in context, as there is no easy comparison with other nodes or even with each node past history. Also, it is impossible to know how much it affected the global network.

Aggregated metrics

The analysis of the performance data may be improved by using some aggregated data like the one showed in the box-plot graphic for the delay (Fig. 4-10). It presents values for the median (central line), 25th and 75th percentile (box), and outliers (plotted

individually). Although there is a high variability in individual data, the 75% percentile for all nodes only slightly exceeds 1 s.

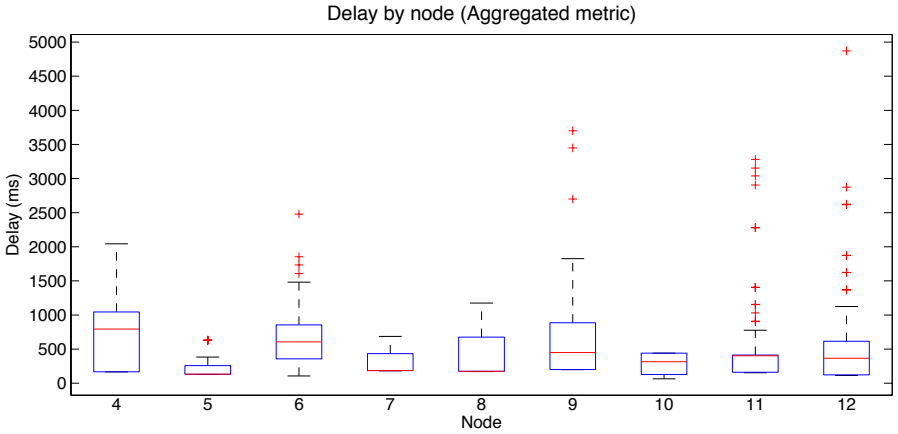


Fig. 4-10 - Delay by node using aggregated data.

Another example is the total energy spent (Fig. 4-11) measured for all nodes.

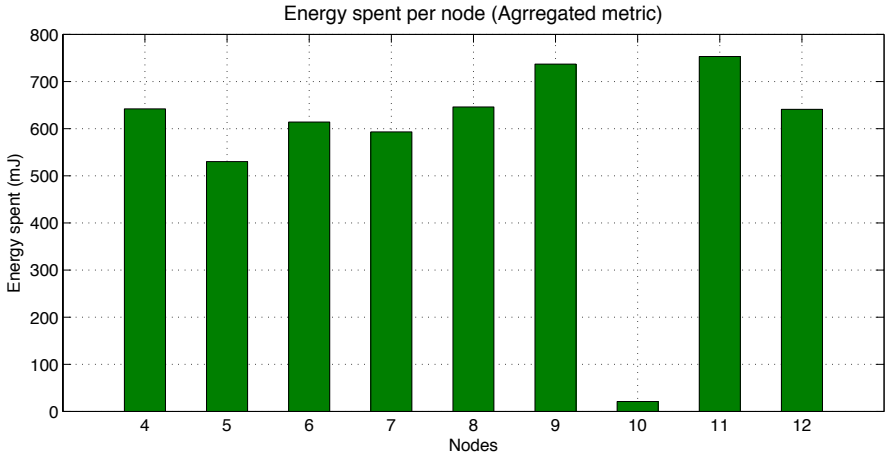


Fig. 4-11 – Total energy spent by node during the simulation.

All the pictures presented show aggregated data from each individual node, during all the simulation time. For all the simulation time node delivery reliability was also measured (packets received divided by the packets sent). It was found that the node delivery reliability ranged from 75.2% (for node 12) to 100% (achieved by node 10, where all packets were received till the node died). The energy spent per message sent ranged from 5.24 mJ of node 12 to 7.25 mJ of node 10.

Statistical data resulting from aggregation can cover all the simulation time or a specific period of time, being able to provide a constant analysis along operation time. It provides a closer analysis of each node performance, especially by comparing it with other nodes at the same level. The aggregation also enables the hiding of small inconsistencies so often in WSNs (such as the imposed loss of 3 packets for node 8), as seen in previous individual metrics figures. However, it does not give an easy overview of the all network as it still focus in individual nodes.

Collective metrics

The use of collective metrics enables a collective analysis of the all network or of a group of selected nodes. In this case, a collective analysis of all the leaf nodes of the WSN tree was accomplished. This view can be compared to individual node's aggregates or to individual node's metrics, depending on the analysis.

Fig. 4-12 shows a moving average of the delivery delay ('Mid' line) of nodes 4 to 12 (leaf nodes of the first scenario) that hides some of the variability that each individual node normally presents (compare to the individual data depicted).

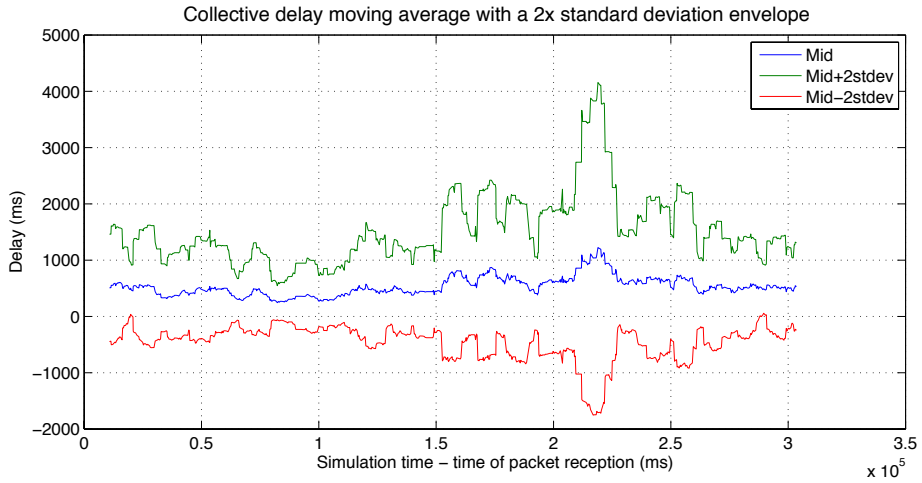


Fig. 4-12 - Collective delay 20 periods moving average, with an envelope of 2 standard deviations.

By using a bigger number of samples, the averages can be further smoothed eliminating temporary hazards, while the exact number of periods to consider depends on the specific QoSensing needs. To improve the analysis, one may also use standard deviation to establish upper and lower bounds to the observation. In this case, a moving average of 20 and an envelope of 2 StDev were used to establish an healthy range for normal network behaviour as well as to show some timely tendencies of the metric. The Collective Network Reliability found for the network during all simulation time was of 86% and the average energy spent per message sent was of 6.10 mJ, which compares directly to the aggregates from individual nodes calculated earlier. This analysis could also be done for smaller time intervals.

While a global view of the network performance can be accomplished by using collective metrics, individual problems are not easy to detect. Events such as the death of node 10 or the individual lost of packets are hidden in the data from other nodes, at least while they do not affect the global network performance tendency. In spite of a small loss of packets may be assumed as normal, the death of a node is not and must be detected. On the other hand, the interference created by using exterior nodes is clear, especially the one from simulation time 150 s to 250 s, as it affects the global performance.

Collective metrics with alerts

To enable the monitoring of individual node problems while using collective metrics to assess for the global WSN performance, another approach was taken. This approach relied in using collective metrics together with alerts. In our simulation three alerts triggers were created: a maximum tolerated time between consecutive packets from the same source, a maximum number of consecutive packets that arrive with delays above healthy values and a minimum energy level. These alerts were the ones found to enable the early detection of network problems, and were chosen according to the requirements stated as more important for WSNs with controlled performance. The first alert trigger was established as the maximum time the WSN application in use can tolerate before the delay can cause a problem – in our simulation was set as 6 seconds. In the second trigger, two values were set. The maximum number of consecutive packets that can arrive with delays above healthy values was set to 3, after some previous simulations. It means that after 3 packets arriving out of the pre-determined healthy envelope, an alert is triggered. The healthy envelope was set to be of 2 StDev around the selected moving average, which was established after simulating the network without any interference. The last trigger corresponds to a minimal energy level that the node needs and corresponds to 10% of its full charge. The results are depicted in Fig. 4-13 (for perception reasons only a part of the simulation time was depicted), and consisted in the trigger of 33 alerts. In this first simulation only individual alerts were tested. All calculations were still done at the sink. By using the specified alerts, Node 10 was found to be missing 6 seconds after the last packet received (our specified limit). The other 32 alerts also referred to consecutive missing reports from nodes, but in this case due to temporary problems that triggered the maximum time allowed between nodes. No consecutive packets from the same source were found to be out of the healthy envelope, nor alerts related to power were triggered.

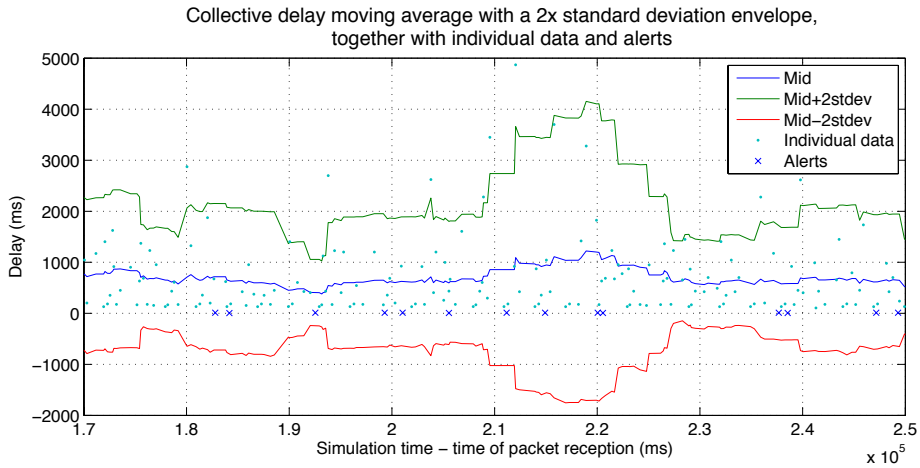


Fig. 4-13 - Collective delay 20 periods moving average, with an envelope of 2 standard deviations, together with individual node data and alerts (selected part from the simulation).

By moving the alert triggers and the collective metrics calculation from the sink to each intermediate node, further improvements can be achieved in the network. Simulations considering both leaf nodes individual performance packets sent to the sink and collective metrics with alerts generated along intermediate nodes, in each scenario described initially - with 13, 22 and 31 nodes, showed that the latter reduced the number of packets transmitted in the all network respectively to 60%, 73% and 55%. The lower reduction was observed in the scenario with 22 nodes and the maximum in the scenario with 31 nodes, as the latter has one more level where aggregation of performance data is possible. The analysis of intermediate Node 1 showed a decrease of the number of packets sent to 33% in the first two scenarios and to 17% in the last scenario when comparing the two kinds of metrics used. The energy spent by Node 1 was also measured and is shown in Fig. 4-14. As seen before, the differences are bigger when more aggregation levels are possible.

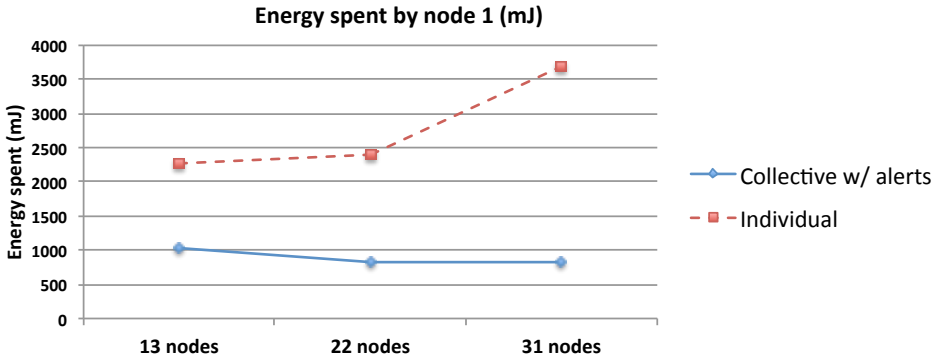


Fig. 4-14 - Energy spent by node 1 using both individual metrics and collective metrics calculated in intermediate nodes, in different scenarios

4.6.4 Evaluation analysis overview

While each type of metric has its own advantages and applications, from the simulations done, and regarding the WSNs with controlled performance scenario with its specific requirements, it can be observed that there are advantages of using collective metrics when assessing the overall QoSensing of a WSN. The use of collective metrics provides a clear overview of the network delay tendency, of the global reliability and energy consumption, which can always be further detailed recurring to the other types of metrics. Also, they can provide the network manager with the information necessary to a real-time global analysis of the network, minimizing the variations and false alarms that result from reading individual data, or data from only one source. Furthermore, they can cover different periods of time or only data related to specific events that may involve a varying number of nodes, being able to provide a constant analysis of part or of the all network, along operation time. When collective metrics are used together with specific alerts, individual events can also be detected, providing for a better and easier monitoring of the QoSensing of the WSN. The monitoring impact can also be drastically improved when

using collective metrics with alerts calculated in intermediate nodes, which is more significant as the number of nodes where aggregation is possible increases. Whenever an anomaly is detected, other metrics, either individual or aggregated, should be used for the needed debug.

In conclusion, the use of collective metrics in WSNs with controlled performance, such as the ones presented in industrial scenarios, enables a global and accurate understanding of the overall performance, can significantly reduce the overhead implied in the QoSensing monitoring and improve its scalability, while not compromising any results. Also, by composing values, small inconsistences disappear, some fluctuation of values is permitted and it easier to detect trend changes in the network.

4.7 FRAMEWORK FOR EVALUATING THE QOSENSING OF A NETWORK WITH CONTROLLED PERFORMANCE IN AN INDUSTRIAL SCENARIO

While performance metrics may be applied to all kinds of WSNs, using any type of application, they are most needed in WSNs with controlled performance. One of the most demanding scenarios is the industrial.

In this section, the work done under the project GINSENG and the know-how obtained, the general set of metrics proposed for the Network performance group of the taxonomy presented in Chapter 3, and the evaluation made to assess the benefits and problems of using different types of metrics, will be joined to create a new framework. This framework targets the operation phase of the life cycle of the network and pretends to be the minimum set that enables the assessment of the QoSensing of a WSN with controlled performance in generic industrial environments (only requirements of the Network performance group are addressed).

4.7.1 Monitoring requirements

One of the most demanding scenarios for WSN with controlled performance is the monitoring and control of industrial processes. Researches made during the GINSENG project, using an oil refinery as test bed, proposed four priority requirements [21]. All of the requirements belong to the Network performance branch of the taxonomy presented in Chapter 3 (Table 4-7).

TABLE 4-7 – NETWORK REQUIREMENTS TO CONSIDER

Delivery delay
Message delivery reliability
Fault tolerance
Energy consumption / Energy efficiency

For an effective monitoring of these networks, performance data must be collected from all nodes belonging to the network, what can be translated in a need of constant feedback.

To enable the performance monitoring of these networks, considering the requirements defined, a new set of requirements emerged. These requirements are presented in Table 4-8.

TABLE 4-8 – REQUIREMENTS IN OPERATION TIME

Reduced overhead associated to performance measurement
Detect packets delay above predefined limits
Detect loss above predefined limits
Detect node's failure in a maximum limit of time
Detect if a node is in critical energy status
Detect if a node is wasting too much energy

The first requirement implies that different metrics (such as delay, loss, reliability and energy spent) are evaluated to assess the overhead implied by adding performance monitoring capabilities to the network. This requirement depends on the rhythm of collection, on the collection protocol and on the specific needs of the application. The requirement to detect the maximum time to detect a node's failure is included in the fault tolerance metrics. All other requirements correspond to thresholds to be put in the metrics that will measure the initial requirements listed in Table 4-7.

Depending on the applications used, the set of requirements presented may prove not to be enough. If the QoSensing necessities of a specific application, running in an industrial environment, include specific security or information efficiency, a reduced set of metrics cannot respond to the requirements. However, the goal of this proposed framework is to select the minimum number of the requirements that are common to a generic industrial scenario.

4.7.2 Proposed framework

A generic framework for enabling a WSN with controlled performance in industrial environments, is composed of 4 parts:

1. A reduced set of metrics;
2. A set of alerts based on the values of metrics;
3. A performance collection protocol;
4. A monitoring tool that triggers corrective actions and where a deeply analysis can be performed (Fig. 4-15).

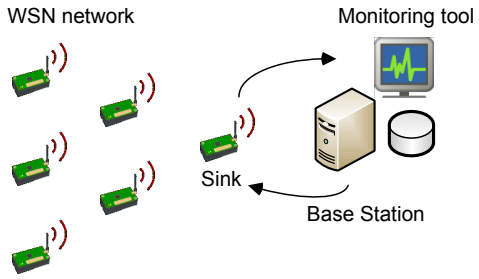


Fig. 4-15 – WSN and monitoring tool.

In what respects metrics, our concern in this chapter, the first two items will be addressed.

A reduced set of metrics and alerts for WSN monitoring, adapted to the assessment of performance in operation time and targeting the priority requirements of a WSN in industrial scenarios, is proposed in Table 4-9. It uses different types of metrics together with alerts to minimize the performance related traffic and focus in the network operation phase.

TABLE 4-9 – METRICS AND ALERTS FOR ASSESSING QOSENSING IN AN INDUSTRIAL SCENARIO

Sink	**Inferred metrics (Passive measurement taken from any packet received)**	**Direct metrics received (From active measurements)**
	▪ One-way node delivery delay; ▪ One-way node loss; ▪ Query delivery delay;	▪ One-way sink delivery delay; ▪ Node (data) delivery reliability; ▪ Collective network reliability (data); ▪ Number of active nodes;
Common nodes	**Alerts generated and sent to sink**	**Metrics calculated and sent to upstream nodes**
	▪ Above threshold hop delivery delay; ▪ Time of last message received from downstream neighbour above threshold; ▪ Node (child) delivery reliability below threshold; ▪ Average energy spent per message sent above threshold; ▪ Critical energy level (below threshold);	▪ Average collective delay per hop; ▪ Average collective energy in nodes; ▪ Total collective packets sent; ▪ Total collective data packets sent; ▪ Total collective packets received;

A monitoring tool connected to the sink, running without processing, memory or energy restrictions will enable the calculation of more complex metrics, including the ones that use network history data. Also, in this framework, no alerts or metrics concerning the actual values sensed are included as they belong to other branches of the taxonomic tree. Only the Network performance branch of the taxonomy is addressed. The thresholds used in alerts must be predefined at deployment and, if necessary, be dynamically changed during operation time.

4.7.3 Framework discussion

Our framework contains the basic metrics and mechanisms to enable a low overhead continuous monitoring of a generic industrial scenario. The fact that it is aimed at

industrial scenarios does not imply that it cannot be used in any other WSNs. In fact, the focus in industrial scenarios relies in the requirements that led to its development.

To respond to the requirements, specific metrics (from the set proposed in Table 4-4) and alerts were specified. Collective metrics were selected to provide for global assessment of performance without implying a profusion of individual packets in the network. These metrics are used in "metrics calculated and sent to upstream nodes" (Table 4-9). Collective metrics evaluate each node together with its children in delay per hop, energy, packets sent, data packets sent (non-performance data packets sent by the node) and packets received. The total collective data sent metric is essential so that the sink can calculate the reliability of the data messages received (messages with actual sensed values).

Together with that metrics other metrics can be inferred in the sink, either by using common data packets as performance packets ("Inferred metrics" in Table 4-9). The sink, using either the collective metrics received and the inferred, can calculate additional metrics that assess the reliability of the network and the number of active nodes (to respond to the fault tolerance requirement).

Alarms are set so that malfunctions can be transmitted to the sink where a deeper analysis can be done or triggered. The metrics associated with the alerts should use specific packets to be sent as quickly as possible to the sink.

By receiving collective metrics at regular time intervals, the sink (or the base station that is collecting the metrics) can perceive the global QoSensing of the network, measured by those metrics. At the same time, it will receive alerts whenever a value exceeds thresholds. Also, two levels of response to a malfunction and two levels of analysis are possible. By calculate metrics in the nodes, they can detect if any problem exists. As an example, if a node detects its energy in low levels it can inform the sink and by itself reduce the duty-cycle. In the base station, not only a deeper analysis can be performed on apparently normal data, based on the collective metrics and on historical data, but also corrective actions may be triggered on receiving an alert.

The base station connected to the sink could also evaluate part of our specific set of requirements in Table 4-7 (if energy is excluded) by analysing a continuous flow of data packets coming from each node. This solution while possible in theory would imply that all nodes (even those that have no data to report) had to send a data packet from time to time with consequences in the amount of energy spent by nodes and in the interference caused by the transmissions. Also, in a hop-by-hop network, upstream nodes would have to forward all those messages to the next level. A deeper analysis on the consequences and trade-offs of this two scenarios will be addressed in Chapter 5.

4.8 ACTING DYNAMICALLY IN THE NETWORK

Message delivery reliability, as seen before, is one of the main concerns when using WSNs with controlled performance. While during the initial deployment of a network all tests may assure (in theory) that the QoSensing of the network will be fulfilled, the dynamic nature of these networks makes that scenario very difficult to endure. In some situations the use of wireless reprogramming is a solution. By reprogramming nodes, it is possible to change the way they work, add new functionalities and adapt the network to different scenarios and situations. However, when changes are more frequent, reprogramming is not an option. Both the time and waste of energy implied in the reprogramming of a node does not make it feasible to use it on a regular basis, to comply with the dynamics inherent to the nature of WSNs.

One of the main causes of low reliability is interference. In a world where wireless devices are growing at high rates and where many of them use the same available unlicensed radio spectrum known as the *Industrial, Scientific an Medical* (ISM) band, interference is a constant item do address [91]. Furthermore, other devices such as remote controls, Bluetooth devices, cordless phones, microwave ovens, also use this band. Even in restrict industrial scenarios, where interference is kept as low as possible, electrical machinery, existing Wi-Fi networks or wireless sensors contribute to the creation of multiple sources of interference. In this section, specific metrics are used to detect interference and specific algorithms are tested to provide for an active response from the network in the form of a dynamic reallocation of channels.

4.8.1 DynMAC and GinMAC

While the theme of network coexistence is out of the scope of this thesis, some work was done in applying some WSN metrics to contribute to reduce its impact. This work was done in collaboration with other researchers, both from the network coexistence area and from the GINSENG project, and resulted in the paper “DynMAC: A resistant MAC protocol to coexistence in wireless sensor networks” [92].

Employing techniques from cognitive radios in WSNs is still a challenge, due to the restriction presented by these networks. A cognitive approach to WSNs must take into account the specificities of these networks and use simple mechanisms. This was the approach taken when creating *DynMAC* (Dynamic MAC). DynMAC, which was built over *GINSENG MAC* (GinMAC) protocol [93], uses dynamic channel reconfiguration mechanisms in order to choose the best communication channel for the WSN, improving the reliability of GinMAC by recurring to the same general properties found in cognitive radios (sensing, decision, sharing and mobility of spectrum).

GinMAC [93] was developed in the GINSENG project as a TDMA MAC layer that provides reliable and timely communications in industrial environments. It also includes a

specific topology control and implements routing (see Section 2.5.3). As a drawback, in GINSENG, GinMAC was limited to 25 nodes per tree. More nodes would imply more time-slots, what was incompatible with the time restrictions that were set. While some mechanisms exist in GinMAC to reduce the loss of packets in presence of interference it does not have any mechanism to avoid interference or enable the coexistence between networks. One of the mechanisms used by GinMAC relies in the *Clear Channel Assessment* (CCA) used in CC2420 radios. In spite of detecting if the medium is busy, before any data is transmitted, this method is not efficient do deal with the constant interference from a noisy environment.

In order for DynMAC to work, the characteristics provided by GinMAC must be available. Specifically, the sink node must know all nodes in the network and their hierarchical position, and it is assumed that nodes only communicate with their direct parent or children, establishing a hop-by-hop communication from the common node to the sink. Broadcast messages from the sink also travel hop-by-hop in the network and are forwarded until reaching leaf nodes. The sink node assures the synchronization of the network by creating a super-frame that attributes each node in the network specific time-slots. Every node has a specific time-slot to transmit, receive and acknowledge packets.

GinMAC details are described in [93]. An introduction is also done in [92].

4.8.2 Interferences between WSNs and wireless local area networks (WLANs)

As IEEE 802.15.4 based WSNs operate in the same frequency band as WLANs (IEEE 802.11) interference occurs between these two different networks. The overlap in the frequency range is clear in Fig. 4-16. This greatly affects their performance, contributing to a low reliability and high packet losses [94].

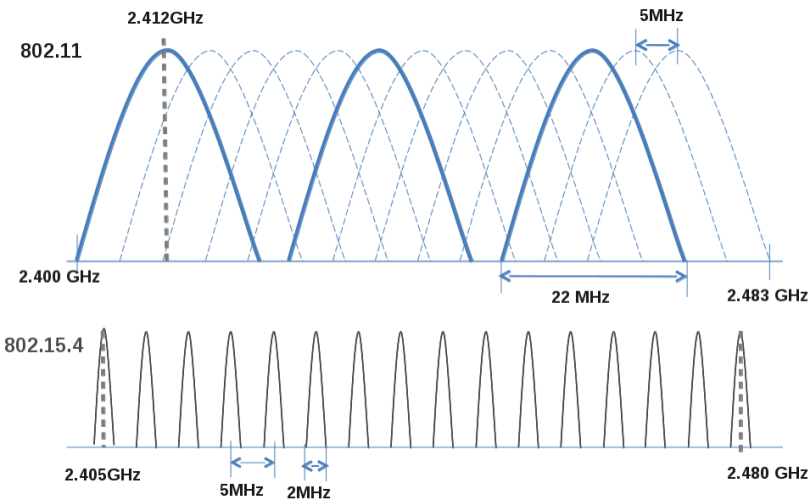


Fig. 4-16 – Frequencies used by IEEE 802.15.4 and IEEE 802.11 [92]**.**

4.8.3 DynMAC protocol description

Several metrics exist that address the quality of the received signal in a node. Among them, RSSI, *Signal to Noise Ratio* (SNR) and *Link Quality Indication* (LQI) are commonly used, being standard in many WSN radios. As SNR is highly correlated with RSSI, and as LQI presents different implementations and also depends on RSSI, a simple RSSI mechanism was chosen.

Network synchronization

In order to start the WSN operation both sink and nodes must communicate in the same channel. So, the sink starts by selecting the local best channel based on RSSI readings and then broadcasts the channel information to the network. Every common node is, at this phase, scanning all channels to detect a broadcast from a specific sink node. On receiving it, they can join the network. Both processes run in the MAC layer.

Selection of the local best channel

A process that runs at the sink makes the selection of the best channel. In the beginning, the selection of best channel is the selection of the local best channel, as the only information that exists is the local information gathered by the sink. Common nodes are, at this phase, only listening. To make the evaluation of the local best channel, both RSSI values and the number of RSSI values above CCA threshold metrics were used. However, as RSSI values translate the signal strength of received packets and no packets are being transmitted, some adjustments had to be made. In fact, when no packets are being received, the RSSI value of each channel indicates the strength of noise and interferences. So, the best channels are the channels that have a lower RSSI and a lower frequency of RSSI values above the CCA threshold. In order to assess the local best channel the spectrum is sensed multiple times, accumulating the values of RSSI and the number of times RSSI is above CCA threshold (Fig. 4-17).

```
                                -- Algorithm 1 --
 1   function sink_local_best_channel()
 2   for(i in 1 to N) do
       /* N: number of times the spectrum is sensed */
       /* Scan of channels 11 to 26 (16 channels) */
 3     for (c in 11 to 26) do
         /* Set c as the current channel */
 4       set_channel(c)
 5       current_rssi = get_current_rssi()
 6       rssi_values[c-11]+= current_rssi
         /* Detect RSSI values above threshold value */
 7       if (current_rssi > CCA_THRESHOLD) then
 8         rssi_above_threshold[c-11]++
 9       end if
10     end
11   end
     /* Sorting accumulated rssi in ascending order */
12   acc_channel_index = index_sort(rssi_values)
     /* Calculates channels cost and gets the best channel */
13   channel_index = cost_computing(acc_channel_index,rssi_above_threshold)
14   best_channel = channel_index[0] + 11
15   return(best_channel)
16   end function
```

Fig. 4-17 – DynMAC: Algorithm 1 – Selection of the local best channel [92].

The procedure scans channels 11 to 26, for *N* times (*N* is an empirical value), collecting from each one the values of the RSSI detected and the number of times that RSSI is above the CCA threshold. These values are accumulated in arrays where each position corresponds to a different channel (lines 6 and 8 of Algorithm 1). Then, the array of RSSI values is sorted and the corresponding list of channels, that has in the first position the channel index with lower accumulated RSSI, is returned (line 12). To join this information with the frequency of CCA above threshold the function *cost_computing* is used (line 13). The pseudo-code of *cost_computing* is shown in Algorithm 2.

```
                                -- Algorithm 2 --
 1   function cost_computing()
 2   cost = 1
 3   for(i in 0 to 15) do
       /* Get the channel at index i */
 4     channel = acc_channel_index[i]
 5     channel_cost[channel] = cost
       /* add the frequency as additional cost */
 6     channel_cost[channel]+=rssi_above_threshold[channel]
       /* computing the cost based on accumulated RSSI for next channel */
 7     if ((i<15) &&
           (rssi_values[acc_channel_index[i+1]] > rssi_values[channel])) then
 8       cost++
 9     end if
10   return index_sort(channel_cost)
11   end function
```

Fig. 4-18 – DynMAC: Algorithm 2 – Calculate the cost of a channel [92].

The cost of each channel is the sum of the cost of its position in the sorted array of indexes (line 5) added with the number of times it was found to be above the CCA

threshold (line 6). The first is obtained by adding 1 for each position in the sorted array of indexes, starting with 1 for index 0 and incrementing 1 for each next position (only if that position has a higher RSSI value, if the value is the same the cost is not changed – line 7). In the end, a sorted list of channels cost, having the lower cost in the first position, is returned (line 10 of Algorithm 2). Index 0 of that list is returned from Function *sink_local_best_channel()* as the best local channel (lines 14-15 of Algorithm 1).

After finding the best local best channel, the sink sets it as the new communication channel and broadcasts the information to the network.

Network joining

In order to join the network, common nodes have to change their communication channel to the one used by the sink. To accomplish it, Algorithm 3 is used (Fig. 4-19).

```
                                -- Algorithm 3 --
 1   function ScanningChannel()
 2   Ci = 11
 3   while (true) do
 4      set_channel(Ci)
 5      if (is_exist_frame( )) then
           /* If a frame exists in channel */
 6         get frame( )
 7         if (is_valid_frame( )) then
              /* Frame ∈ sink's GID */
 8            break
 9         end if
11      end if
12      if (Ci < 26) then
13         Ci++
14      else
15         Ci=11
16      end if
17   end
18   return (Ci)
19   end function
```

Fig. 4-19 – DynMAC: Algorithm 3 - Detecting the communication channel of the sink [92]**.**

By running Function *ScanningChannel()* common nodes repeatedly scan the channels aiming to detect a frame sent by the sink. If a frame is detected (line 5) in one of the channels, its validity is checked by analysing if its *Group Identification* (GID) is the same as the GID in the node (line 7). If it is, then the frame is assumed as coming from a valid sink and the scanning process is stopped. If no frame is detected or if the detected frame is not valid, the node keeps changing channels until it finds the sink communication channel.

After finding the sink's channel, the node sets the radio to the one used by the sink and initiates the joining process by sending a joining frame to the sink. On receiving a joining frame from a node, the sink saves the information and sends an ACK frame to the node. On receiving the ACK the node knows that it has joined the network.

Global best channel selection

The fact that common nodes were able to join the network does not mean that they are using the best channel, as the initial selection was only based in the local information of the sink, not considering the interferences and noise that may occur in common nodes for that channel. In order to find the global best channel another phase is necessary. Ideally, this second phase should only take place after all nodes join the network. However, as it is not always possible, this phase starts if at least one node besides sink has joined the network for a predefined amount of time.

The global best channel selection process can be split in 8 steps (Fig. 4-20):

1. The sink broadcasts 3 messages soliciting channel information from all other nodes.
2. On receiving the soliciting message from the sink the common node forwards it to its children and samples the RSSI in all channels using Algorithm 1 (already used by the sink to choose its local best channel).
3. Each common node uses Algorithm 2 to compute each channel cost.
4. Each common node sends an ordered list of its channel cost to the sink.
5. After having the information from common nodes, the sink uses Algorithm 4 to decide which is the global best channel.
6. If a new global best channel is found (different from the one being used), the sink broadcasts 3 switching messages to ask all other nodes to switch to the new channel.
7. On receiving the switching message from the sink, each common node forwards it to its children and waits for a predetermined amount of time between switching to the new channel (in this way it can forward all the 3 switching messages from the sink increasing the possibilities that every node receives at least one).
8. The sink node switches to the new channel after sending the last switching message.

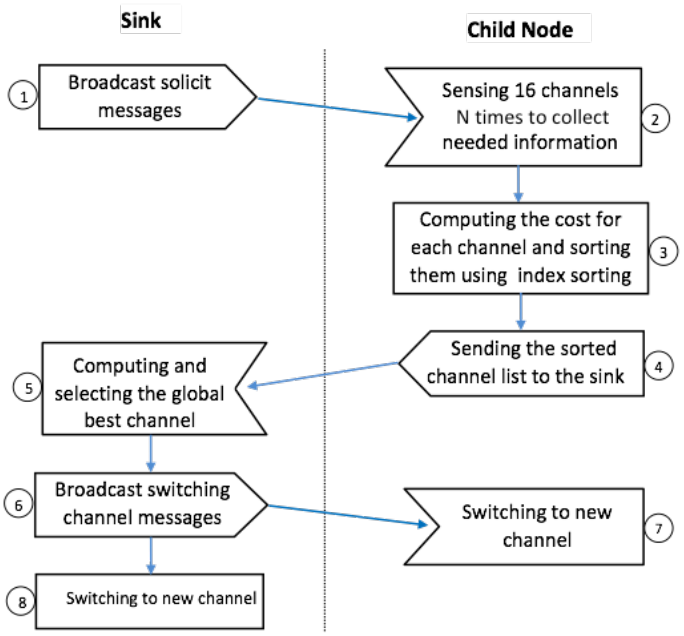


Fig. 4-20 – DynMAC: Choice of the Global best channel [92].

The process used by the sink to choose the global best channel is shown in Fig. 4-21.

Algorithm 4 – Choosing the global best channel

```
1   function GlobalBestChannel()
    /* Compute the cost for each channel */
2   for (node_id in nodeList) do
3      for (pos in 0 to 15) do
          /* Get the channel at pos */
4         channel = Channel_index_list[node id][pos]
5         channel_cost[channel]+=pos+1
          /* Store the worst channel list */
6         if (channel_index_list[node_id][channel] == 15) then
7            Bad_channel[channel]+=1
8         end if
9      end
10  end
    /* Sorts channels by cost and store in channel index global */
11  Sort_channel(channel_cost, channel_index_global)
    /* Choosing the best channel */
12  best_channel = channel_index_global[0]
    /* Reselect the best channel if the current best channel exists in
bad_channel list */
13  c=0
14  while ( (c < 16) && (bad_channel[c - 1] > 0) ) do
15     c++
16     if (c < 16) then
17        best_channel = channel_index_global[c]
18        break
19     end if
20  end
21  return best_channel
22  end function
```

Fig. 4-21 – DynMAC: Algorithm 4 – Choosing the global best channel [92].

Having received a sorted list of channels from every node (ordered from the best in the first position to the worst), the sink starts to calculate the total cost of channels in the network. To accomplish it, the value of the position of each channel in the sorted list received from each node, is added to an array of channels cost (line 5). That array will contain the sum of all the individual costs of each channel, from each node. Specifically, for each node, the cost of a channel will be 1 for the channel in the first position (as the array starts with 0 is the position in the array + 1), incrementing one for each next position (lines 2 to 5). To avoid choosing the worst channel of any of the nodes (what would create problems for those nodes), the sink also saves them in a list (lines 6-7). The worst channels correspond to the channel in the last position of every node's list. The resulting array *channel_cost* is then sorted with the best channel in the first position. The channel with the lower position that does not belong to the worst list is chosen as the best channel (lines 14 to 21).

Periodically re-evaluation of the best channel

As the quality of channels may change during the lifetime of a network, a periodic re-evaluation is necessary. To do it, DynMAC uses Packet Error Ratio (PER) calculated from the packets not correctly received by the node (the method of calculation is similar to network reliability but, instead of the packets received, packets not received are used – see Section 4.5). Every specific time interval (DynMAC currently uses the last 5 minutes), each node calculates its PER and if it surpasses a predetermined threshold (currently 1%), a notification is sent to sink. Based on the notifications received from nodes the sink makes a decision on the need for a channel re-evaluation. If the decision is to re-evaluate, an identical process as the used to determine the global best channel is used (Fig. 4-21).

Recovery from lost link

To enable the recovery of a node that looses the connection to the sink (e.g. the node did not receive a switching channel message from the sink), each node is forced to re-scan the communication channel if a communication to the sink is not possible for a certain amount of time. Algorithm 3 (Fig. 4-19) is used to accomplish it.

4.8.4 DynMAC evaluation and analysis

DynMAC was evaluated both by simulation and in a real test bed using a network with 13 nodes (Fig. 4-22).

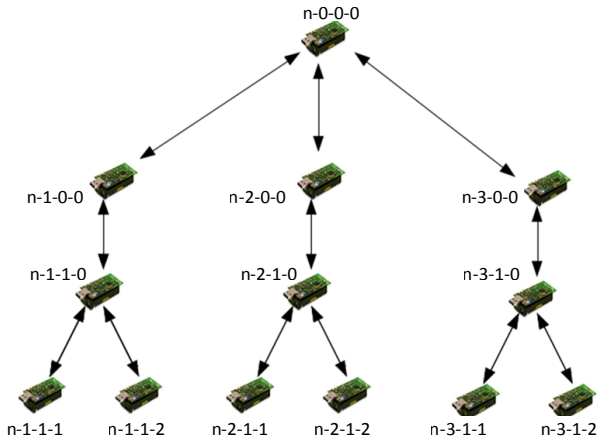


Fig. 4-22 – DynMAC topology used in tests [92]**.**

4.8.4.1 Simulation

The first tests were made using a simulation consisting of a scenario with 13 nodes. Simulation was made using the Cooja simulator [89]. Transmission range was set to 25 m and the distance between each level was of 5 m (all nodes suffer interferences from each other). These values were chosen to test the protocol for in-network interferences. However, the results apply to any scenario. The super-frame was configured to have 100 slots, resulting in a 1000 ms super-frame (10 ms per slot). Each node sent 1000 packets to the sink, at a rate of 1 every 5 s, so that packet loss could be tested. To test scanning and network coverage time, the experiments were repeated 300 times.

Scanning and network coverage time

The scanning test measured the time that each node took to detect the channel of the sink. The network coverage measured the time it took all nodes to successfully join the network. For both tests the time considered starts after the booting of the sink node. Table 4-10 details the results of the tests showing the values obtained by the nodes located at each different level of the network tree.

TABLE 4-10 – DYNMAC SIMULATION – SCANNING AND BOOTING TIME [92]

	Level 1		Level 2		Level 3	
	Scanning (ms)	**Finish booting time (ms)**	**Scanning time (ms)**	**Finish booting time (ms)**	**Scanning time (ms)**	**Finish booting time (ms)**
Minimum	112	160	24	72	24	136
Q1	2228	2652	784	1170	822	1074
Median	3798	4054,5	2104	2352	2236	2468
Q3	4860	5462	2483,75	2753,25	4264	4554
Maximum	8648	8920	8872	8984	14264	14856
Average	3842,68	4175,21	2309,26	2569,97	3056,15	3332

As shown in Table 4-10, nodes at level 1 took an average of about 4 super-frames (that last for 4000 ms) to detect the communication channel of the sink. The minimum time of the scanning took just 1 super-frame (112 ms < 1000 ms) and the maximum was of 9 super-frames. Nodes from other levels took identical times with the exception of some nodes in level 3. After the selection of the channels the nodes need an additional time to finish the booting process, which in most of the cases took less than 600 ms. The Finish booting time is also shown in Table 4-10. The coverage time was always less than 33 s (8920 ms + 8984 ms + 14856 ms = 32760 ms) assuming that each child node would receive the channel information from their parent using the maximum time for each tree level. In all the experiments the coverage time measured was less than 17 s (maximum of 16920 ms).

Handoff time

The handoff time, i.e., the time that the node takes to re-join the network (its parent node) after having left it, varies from 3 ms to 384 ms. If the node is not a leaf, an additional time must be considered as its children also have to resynchronize with it and with the sink. From the simulations it took less than 678 ms for other nodes to resynchronize with their parent.

4.8.4.2 Real test bed

The same scenario used in simulation was setup using TelosB motes. Different Wi-Fi networks in range produced interferences, which resulted in different channel qualities.

Evaluating the quality of the channels

The first test evaluated the quality of the IEEE 802.15.4 channels using Algorithm 1 described in Fig. 4-17. The test was repeated 1500 times and the number of times a channel was considered the best and the worst was counted (Fig. 4-23). The reason why channels 25 and 26 had the highest probability of being chosen is because these channels are out of the IEEE 802.11 WLAN spectrum, the main interference that existed for this experimental setup. The probable reason why the best channel changes is related with the dynamic nature of the interferences and because of the periods where the WLANs have no traffic. If the best channel is scanned in one of these periods, a channel may be considered as the best channel. Channels 23 and 24 were identified as the worst ones most of the times what is explained by the existence of a WLAN working on channel 11 of IEEE 802.11g (see allocation frequencies in Fig. 4-16).

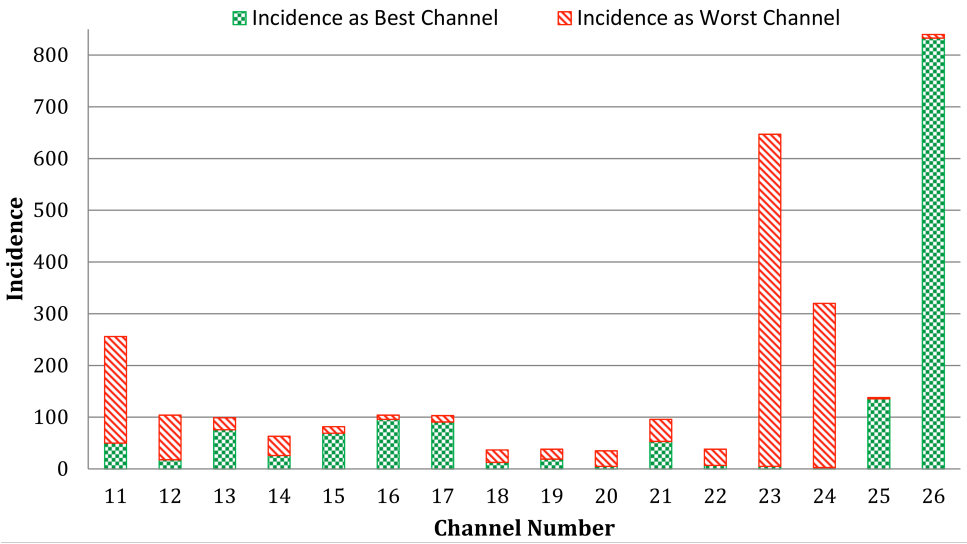


Fig. 4-23 – DynMAC real test bed: local best and worst channels [92].

Scanning and network coverage time

The scanning test measured the time that each node took to detect the channel of the sink as had been previously done by simulation. By analysing the results it is clear that most of the nodes scanned and detected the correct channel within the time of 1 super-frame (<1000 ms). The maximum time to detect the sink channel took was of 2120 ms, the time of 3 super-frames, and occurred in nodes in Level 3 of the network tree.

With a few exceptions, the time to finish booting was similar to the one obtained by simulation. As for the coverage time it can be estimated to be less than 7 s (corresponding to the sum of maximum boot time of nodes in level 1, 2 and 3). Most of the times it took about 3 s to cover the entire network, and the maximum was of 6040 ms (Fig. 4-24).

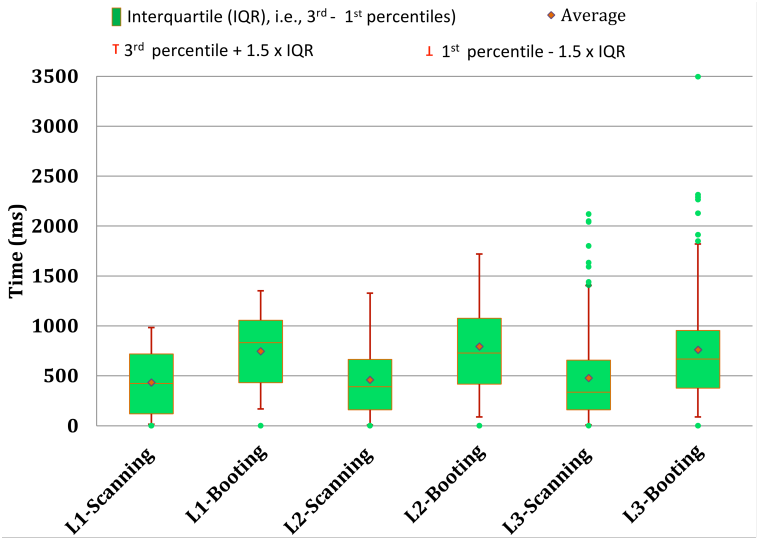


Fig. 4-24 – DynMAC real test bed: scanning and booting time [92].

Dynamic interferences and recovery from lost link

To assess the capabilities of DynMAC to dynamically change channel, interference was created by having 4 external nodes continuously broadcasting in the network channel. The interval for computing the PER was set to 5 minutes and the threshold defined to 1%. If after 5 minutes the calculated PER was above 1% then an alert was sent to the sink. The sink was programmed to start a re-evaluation of the best channel on receiving a minimum of 3 alerts from the nodes. The results were that after a period between 10 and 15 minutes (a node has to wait at least 5 minutes to calculate the PER and the sink needs at least 3 alerts), the sink started re-evaluating the best channel.

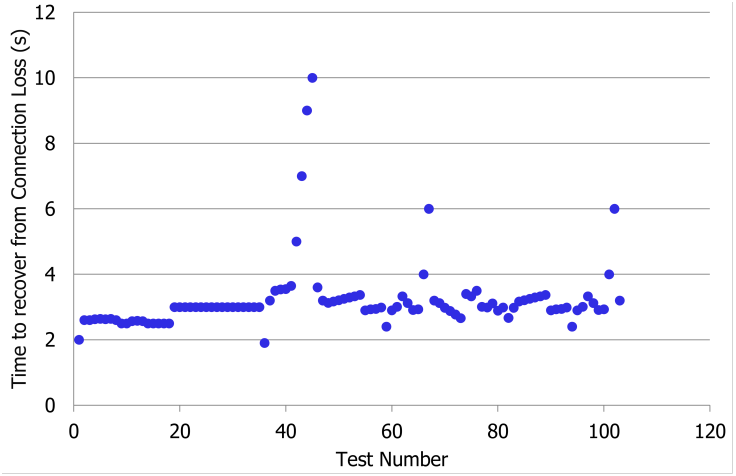


Fig. 4-25 – DynMAC real test bed: recovery from lost link [92]**.**

To evaluate the ability for a node recover from a lost link a switching process was forced and during that time, one of the nodes was removed from the range of its parent. When all nodes successfully switched to the new channel the node was put in its initial position. Both leaf and intermediate nodes were tested. In most of the cases it took about 3 s for a node to resynchronize with the network, with a maximum of 10 s. (Fig. 4-25)

4.8.4.3 Analysis of worst and best channels

To assess the impact of using different channels, the three best and three worse channels of consecutive tests in the real test bed were analysed for their average packet loss ratio. Results showed that for the worst channels the packet loss ratio could be as high as 3% while the best averaged 0.25%. Being DynMAC based on GinMAC, a TDMA protocol that by itself guarantees a reduced loss ratio, the results are significant, showing the improvements that can be achieved by using the best communication channel available.

4.8.5 Evaluation analysis overview

The mechanisms proposed in DynMAC, using as metrics the RSSI and PER, proved to be useful to assess the better communication channel available in the network, when interferences exist. By employing techniques similar to those found in cognitive radios DynMAC adds an additional reliable level to a MAC already targeted for scenarios that demand controlled performance (GinMAC). The mechanisms proposed in DynMAC proved to be capable of dynamically adapt to a scenario where the quality of channels varies as happens in noisy and interference-prone environments. This new mechanism results by enabling the reaction of the network to a changing environment. The response is sustained by the analysis of specific metrics from all the nodes that belong to the network, which provide the sink with the necessary information to act and guarantee a sustainable performance of the network along its lifetime.

4.9 CONTRIBUTIONS TO GINSENG PROJECT

In the context of the contributions made to the GINSENG project by the author of this thesis, an initial set of metrics was proposed to evaluate the priority requirements specified in Section 3.6. An initial proposal was included in Section 5 of project deliverable D1.1 – "GINSENG Architecture, Scenarios and Quality of Service Measures" [25]. The proposal was detailed in Sections 2.5.3 and 2.5.4 of project deliverable D1.3 – "Final GINSENG Architecture, Scenarios and Quality of Service Measures" [21]. The metrics use a MIB, located at each node, where aggregated node and neighbour info are saved [95]. This MIB includes parameters such as total packets sent/received, number of retransmissions, average RSSI, uptime, radio listen and transmission time, and is periodically sent to the sink or retrieved by query.

The results of applying this metrics to the GINSENG laboratory test bed and real test bed in the oil refinery in Sines, were made by other members of the project, responsible for Work Package 4 - "System Demonstration and Evaluation" (see Section 2.5) and are detailed in project deliverable D4.3 – "Software integration (First evaluation phase)" [95] and project deliverable D4.5 – "Second Software Integration and Preliminary Evaluation" [88].

4.10 CHAPTER SUMMARY

In this chapter, different types of metrics were presented and their characteristics and applicability to WSNs was discussed. Collective metrics were introduced as a useful type of metrics to address the evaluation of the global network QoSensing, while saving resources and hiding the normal fluctuation of values in networks subject to many hazards. A set of recommendations to follow when creating metrics for WSNs, considering their special characteristics, was also presented. Next, the performance

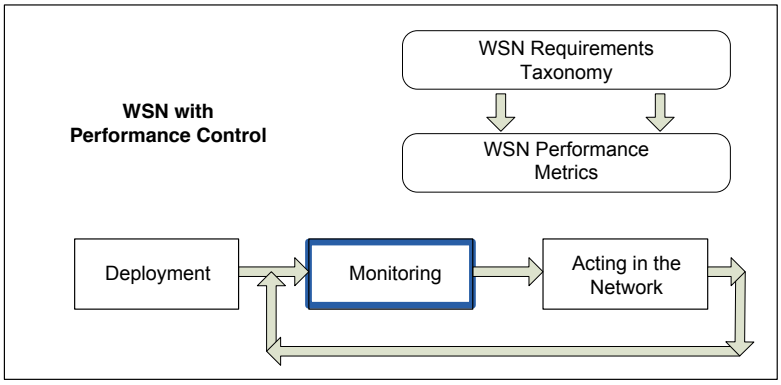
branch of the previously proposed taxonomy was studied in depth and a general framework of metrics was proposed, adapted to each of the phases of the life cycle of a WSN. After evaluating the potentialities of each type of metric in the assessment of the QoSensing of a WSN, a framework for its measurement, targeting WSNs in industrial environments, was proposed. Finally, select metrics from the initial set proposed were used to dynamically change the working channel of a WSN, in order to guarantee that it is working on the best channel possible.

In order to better assess the QoSensing of a network, its metrics should be sent to the sink, where a more detailed analysis can be made. The collection of metrics is the subject of next chapter.

Chapter

5

A Data Fusion Protocol for WSN Performance Collection



Performance monitoring in WSNs with controlled performance implies that a huge amount of performance data is used. Performance collection deals with the mechanisms used to gather performance information and send it from the network nodes to the sink.

In this chapter, the work already done in the area of monitoring of WSNs and some solutions that use composed metrics are presented. Next, data fusion is reviewed and its application to WSNs, especially to the case of WSNs performance monitoring, is addressed. Finally, a protocol for performance data collection that uses data fusion to overcome the profusion of performance packets is proposed and evaluated by simulation.

5.1 INTRODUCTION

The specific capabilities of WSNs, such as fast deployment and flexibility, together with the low cost solutions that can be achieved, bring new opportunities for an all-new range of applications, far from the typical scenarios of low requirements and high redundancy. Some of these new scenarios involve the assurance of predefined performance goals and QoSensing. To guarantee that these goals are achieved, an effective monitoring and control of the network must exist. This monitoring must be done continuously, especially when using critical applications, ideally in real-time, so that immediate responses can be triggered in the presence of anomalies. However, this monitoring depletes nodes energy (especially in middle nodes that have to retransmit the information to the sink), increases interference between nodes and causes an overhead in the used bandwidth, a foreseeable result if continuously measuring performance of a large multilevel WSN. Also, these new WSNs with controlled performance tend to use a careful planning, with nodes strategically located near their sensing target, minimizing the global number of nodes and their redundancy, and minimizing the need of automatic self-configuration. Furthermore, this new networks do not require a specific knowledge of the performance of every node in real-time, as it is naturally subject to variations due to the nature of WSNs. What is required is an effective knowledge of the performance of the global network, or of the performance of distinct groups of nodes, and the assurance that if any node metric deviates from a predetermined range that information is immediately sent to a central base station where further analysis can be made and eventually corrective actions can be triggered. To respond to this new necessity, a protocol that uses data fusion will be presented. It is designed to enhance the benefits of continuous monitoring while minimizing its implied overhead, using the specific characteristics of the WSNs with controlled performance in its favour.

5.2 RELATED WORK

Some work has already been done by the sensor's research community in protocols, mechanisms and tools that perform data analysis and use aggregation operations before transmitting data to next nodes. Next, generic protocols and mechanisms that use aggregation are presented, followed by some existing WSNs diagnostic tools.

Protocols and mechanisms

Directed Diffusion [14] is a data-centric approach to the dissemination of data in a WSN. It works by requesting a set of named data from application-aware sensors. Each set of named data requested is built from attribute-value pairs that specify an *interest*. That interest is sent to the network by the sink. On receiving an *interest* request, a node disseminates it through its neighbours (to all or to selected ones that previously responded to that *interest*), asking that the reply be sent to them, not directly to the sink, promoting local communications. All the nodes that have the data specified respond to the request

using attribute-value pairs to specify the answer. The results are transmitted node by node and can be aggregated (e.g. duplicated data is suppressed) saving energy. The communication is made between a local source, which requests information, and a destination that has it, in a non end-to-end transmission, preferring short-range hop-by-hop operations to end-to-end long-range communications. It performs computation over data to reduce data transmission and obtain energy savings. Directed diffusion is not suited to networks with a permanent broadcast of data, nor uses specific aggregates for each type of data, being more useful in a query-driven model.

TAG [96] (Tiny AGgregation) is a generic aggregation service for ad-hoc networks. It provides a declarative interface for data collection and aggregation inspired by database query languages and distributes those queries in the network. It has two different phases, a distribution phase and an aggregate phase. In the first, the request is sent to the network where nodes propagate it to their children and then wait a certain amount of time for their response. After receiving the data in the specified time (all data must be within a specific time interval, called epoch, that is sent in the query), it uses the aggregation function specified by the initial query to aggregate the value received with its own values and sends it upstream to the sink. This protocol is suited for generic data and only works in a query-driven data-delivery model.

In [97] Zhao et al. describe an architecture for sensor network monitoring and propose a protocol to continuously compute aggregates of network properties (e.g. loss rates, energy levels, packet counts). The protocol uses *digest* functions whose input is the contribution value received from each node. This *digest* functions use decomposable aggregates, i.e., aggregates that can be calculated from partial aggregates. These *digests* are continuously diffused throughout the all network and allow for all nodes to have a constant knowledge of the network. The protocol targets energy efficiency and uses piggybacked messages sent by other protocols to transport its own data values. It also uses a tree that spans the entire network to compute specific *digests* and avoid duplicates. The proposed architecture uses 3 levels of monitoring, each using different tools. The first dumps detailed node status for network diagnosis and is only used when there is a strong belief that a node has problems. The second uses *scans* that spread over the network to ask for a certain network property value. The values returned are aggregated along the network if their values are similar. After detecting the area of the problem, dump may be used. The last level of monitoring is the continuous *digest* of network properties. When some of the computed aggregates indicate that there is a problem somewhere in the network, other levels of monitoring are invoked. The computation of *digests* does not use any centralized approach or head nodes. Instead, each *digest* function computes its part of the function and sends it to the node's neighbours. Eventually, using a tree computed using some of the *digest* functions, all that info reaches the root of the network. In spite of saving energy and allowing for a general knowledge of specific network properties, the proposed approach is not indicated for critical applications as it relies on changing spanning trees,

does not deliver data towards a base station that may take corrective actions nor has any alert mechanism in case a critical value is detected in a specific network property.

WSNs diagnostic tools

Some tools were also created to diagnose WSNs. These tools can be divided in active and passive, with some tools combining the two approaches. Active monitoring implies that the nodes gather and transmit specific performance data using their own resources. This is accomplished by having specific software in the network nodes. By using passive monitoring a new network is deployed that overhears the one to be monitored, therefore not interfering in any way, nor wasting any of its resources (although some interference may arise). As the use of passive monitoring implies the existence of a second network its use may be limited to the initial deployment phase.

Sympathy [98] is an active monitoring and debugging system in which sensor nodes are supplied with specific monitoring software, which periodically sends local node metrics to a dedicated sink node. The mechanism developed is aimed at detecting and debugging failures in sensor networks and is specifically designed to data gathering applications. Sympathy uses selected metrics to enable failure detection and includes an algorithm that root-causes failures and localizes their sources in order to reduce overall failure notifications and point the user to a small number of probable causes. Metrics are collected in three different ways, by Sympathy code running in nodes that actively send information to the sink, by monitoring application and transport traffic in the sink, in order to discover metrics passively, and by having Sympathy code (running in the sink) extracting information from the sink application itself. Sympathy uses three kinds of metrics categories: connectivity, flow and node metrics. The first contains the node's routing tables (sink, next-hop and path quality) and neighbour lists. The second includes packets transmitted and received by each node. The third has the node's uptime (to detect reboots), and node's good and bad packets received (to detect congestion). Active metric collection is done in a predefined interval. Passive metrics result in the assumption that, in a data-collection network, there is a direct relationship between the amount of data collected at the sink and the existence of failures in the network. So, thresholds of packets delivery are measured in order to detect possible failures. The advantage is that these metrics do not add resource costs to the network. However, in the presence of active metrics these metrics are redundant. All evaluation is done at the sink, the only node that offers no resource constraints. In the sink, normal and generated Sympathy traffic, are analysed for failure conditions. When a failure occurs, Sympathy triggers failure localization and reporting.

The *Sensor Network Management System* (SNMS) [99] is an application-cooperative management system for WSNs. It provides two functionalities, a query system to enable rapid, user-initiated acquisition of network health and performance data, and a logging system to enable recording and retrieval of system-generated events. The first allows

programmers to easily get specified information by running a query. These queries use simplified notation in order not to create any overhead in the network. By allowing queries, SNMS avoids the constant sending of info to the network and places management at user's control. The second enables post-mortem analysis by supporting an event logging system. SNMS is designed to have a lightweight approach to the network, and was developed trying to occupy a minimal amount of RAM, generating network traffic only in response to direct human action, and to generate no network traffic in the steady state. It also aims to be simple and robust, and to depend on the application as little as possible, as to ensure that it will continue in function even when the application fails.

Memento [100] objective is to monitor WSNs health, providing failure detection, symptom alerts and logging, while having low energy consumption and bandwidth usage. All nodes monitor one another to implement a distributed node failure detector, a symptom alert protocol and a logger. It requires nodes to periodically send heartbeats to an observer node (the node above in the routing topology). Those heartbeats are not directly forwarded to the sink node, but are aggregated in form of a bitmask (i.e., bitwise OR operation) that results from the combination of each node local state with the results of its children within the routing topology. Each result summarizes the state of the node's sub-tree (including that node) for the required discrete health symptoms (e.g. failure detectors for local radio neighbourhood, low battery alarms, local radio congestion). The protocol calculates the entire network result every sweep interval.

Sensor Network Inspection Framework (SNIF) [101] is a passive monitoring system that uses a separated deployment-support network. SNIF intends to provide a passive approach for sensor network inspection, by overhearing and analysing sensor network traffic to infer the existence and location of typical problems encountered during deployment. To accomplish this, it uses the concept of *Deployment-Support Network* (DSN) [102], minimizing the changes and impact on the WSN by attaching an additional node to some or all WSN nodes. By creating an additional DSN in the same area where the WSN was deployed, SNIF avoids the direct connection to WSN nodes, and uses the radio receiver of the DSN nodes to create a network-wide distributed sniffer [103]. The aim is to develop a tool for passive inspection of sensor networks, where the network state can be inferred without instrumentation of sensor nodes. In order to detect network problems, SNIF tries to avoid an approach that requires active monitoring software in sensor nodes. In this manner, it does not suffer from the same problems as the network being monitored, what could cause the reduction of the desired benefit. Also, it tries not to use the scarce sensor network resources with inspection overhead. Finally, by simply overhearing the desired WSN, it avoids having to add or remove instrumentation from the nodes, what could lead to subtle changes in application behaviour. SNIF supports the detection of specific problems that occur during deployment of a sensor network by using passive indicators for each of the problems, which allow inferring the existence of a problem from observed packet traces. An indicator is an observable behaviour of a sensor

network that hints (in the sense of a heuristic) the existence of a specific problem. Problems are classified into four classes based on existing deployments: node problems that involve only a single node, link problems that involve two neighbouring nodes and the wireless link between them, path problems that involve three or more nodes and a multi-hop path formed by them, and global problems that are properties of the network as a whole.

Distributed Passive Monitoring in Sensor Networks [104] presents an architecture and a tool (Passive Island Monitoring Tool) to monitor sensor networks, either in development as in operation scenarios, using a monitoring hierarchy that sends information towards a central analysis station. The monitoring proposed is done in a passive way, using monitor nodes placed inside the target WSN. These sensor nodes have two radio interfaces, the first used to overhear the surrounding communications (in order to prevent any influence in the existing network), and the other to transmit the collected information to the next level of the monitoring hierarchy towards a central server where data should be visualized in real-time for further analysis. Hierarchical monitoring implies that multiple deployed monitoring nodes have to be interconnected by a network separate from the original WSN.

Distributed Node Monitoring in WSNs (DiMo) [105] presents a distributed and scalable solution for monitoring the nodes and the topology, along with a redundant topology for increased robustness. The aim is to operate in safety-critical WSNs, where all sensor nodes must be up and functional. In these scenarios, once an event is triggered on a node, the information must be forwarded immediately to the sink, without setting up a route on demand or having to find an alternate route in case of a node or link failure. It provides two functionalities, network topology maintenance and network health status monitoring. The first is accomplished by the maintenance of a redundant topology, based on relay nodes (neighbours that can provide alternative routes to the sink). Each node has a minimum of two relay nodes. Monitoring is done by using observer nodes that check the reception of heartbeats within a certain monitoring time. If the heartbeat is not received, the observer sends a missing node message to the sink. There is always one observer for each node. The sink is always aware of which nodes are being observed in the network, and therefore always knows which nodes are up and running. While most of the existing solutions assume a continuous data-flow from all the nodes in the network, this approach observes the nodes and topology locally, only reporting to the sink if a node failure is suspected.

Passive Diagnosis for WSNs (PAD) [106] is an online diagnosis approach that passively looks for network symptoms from the sink, using probabilistic inference to root causes of abnormal behaviours in the network. It was proposed as a lightweight and scalable network diagnosis mechanism, in opposition to other tools that require the generation of extra data and require high levels of computation and energy. PAD is constituted by four components, a packet marking module, a mark-parsing module, a probabilistic inference

module, and an inference engine. The packet-marking scheme of network data reveals the inner dependencies of sensor networks and avoids traffic overhead while overcoming scalability problems of other approaches. When the inference algorithm, from the available input data, observes specific symptoms a probability of failure is deduced. This work was motivated and tested on a sea monitoring project constituted by a group of nodes that float and retrieve scientific data that is transmitted to a sink node.

Next table summarizes the main characteristics of the selected monitoring tools.

TABLE 5-1 – MAIN CHARACTERISTICS OF THE MONITORING TOOLS.

					Information provided		
	Monitoring type	**Type of preferable application scenario**	**Implies separate network?**	**Hierarchical with data aggregation**	**Failure detection**	**Symptoms alerts**	**Logging for post-mortem analysis**
Sympathy	Active + Passive	Data gathering	No	No	Yes	-	-
SNMS	Active w/ Query Driven	-	No	No	Yes	-	Yes
Memento	Active	-	No	Yes	Yes	Yes	Yes
SNIF	Passive	-	Yes	No	-	-	-
Passive Island Monitoring Tool	Passive	-	Yes	No	-	-	-
DiMo	Active w/ Event Detection	-	No	No	Yes	Yes	-
PAD	Passive	-	No	No	Yes	Yes	-

While useful during the deployment phase of the primary network, the use of passive monitoring implies extra costs and a duplication of the effort in deployment and management, not being adapted for general use. None of the tools provide for a constant reporting of the effective QoSensing in the network except by flooding the network with performance packets. Memento, while aggregating information from all the nodes in the path to the sink in a status bitmask, only reports the generic health of that sub-tree not giving any information about its current QoSensing.

None of the solutions presented is adapted to the needs of a WSN with controlled performance, especially if used in critical environments. In order to effectively monitor a WSN with controlled performance, a solution that provides a constant knowledge of the network health, that triggers alerts if anomalies are detected and with debugging possibilities, while keeping a low overhead, is necessary.

5.3 DATA FUSION REVIEW

In this section, data fusion terms, classifications, architectures and algorithms will be reviewed. The aim is to understand the mechanisms already existent and analyse the possibilities for their application to WSNs, especially in a new approach to a metrics collection protocol.

5.3.1 Data Fusion terms

Data Fusion is used in a variety of research fields such as statistics, control, robotics, computer vision, networking, and by who studies ways to combine data from multiple data sources. Several terms have been used in literature to address the *fusion* subject. The different terms correspond to different views, contexts, architectures, systems, theories or techniques. The most used terms correspond to three different levels of fusion: *information fusion*, *sensor fusion* and *data fusion*. The first corresponds to the highest level, a level where information cannot be directly translated in numbers, and is used in artificial intelligence contexts. The second term is used to specify results from the process of fusion of data collected by sensors, as in WSNs. Finally, *data fusion* is usually applied to the fusion of raw sensor data. Other terms that also appear in literature are *multi-sensor fusion* and *data aggregation*. The first deals with the "*synergistic use of multiple sources of information*" [107], relying on "*redundant or complementary data to maximize the information content and reduce systematic and random errors*" [107]. The second, *data aggregation*, is widespread and sometimes confused with data fusion. Data aggregation deals with the summarization of data from multiple sensors sources, resulting in a more comprehensive, non-redundant and reduced amount of data. However, this summarization may lead to some loss of accuracy of the result data, reason why data aggregation should be avoided as a general term for data fusion. Instead it can be seen as a part of data fusion, the aggregation of data for summarization [108]. A representation of the relationship between several fusion terms is depicted in Fig. 5-1. While each term has its own specific meaning, data fusion is broadly accepted as an overall term, and therefore will be used as such throughout this document.

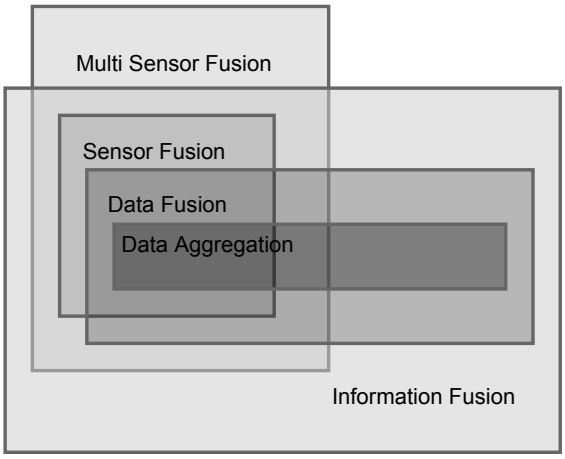


Fig. 5-1 - Relationship between fusion terms (adapted from [108])

The term *data fusion* had many definitions proposed in literature along the years, especially by military and in remote sensing research fields [108]. Not only the term is a generalization of different specific aspects, as has been seen, but also comprises the different applications, techniques or mathematical tools used. The definitions proposed depend on the field of research, and emphasize different aspects that characterize specific fusions. The Data Fusion Workgroup of the US Department of Defence analyses data fusion as a process for the improvement of data quality [109]. Hall [110] also defines data fusion in terms of information quality stating that "*Data Fusion techniques combine data from multiple sensors, and related information from associated databases, to achieve improved accuracy and more specific inferences that could be achieved by the use of a single sensor alone*". Wald [111] focus data fusion by analysing it from a point of view of framework that includes "*means and tools for the alliance of data originating from different sources"*. In [112] Abdelgawad and Bayoumi define data fusion as "*the use of techniques that combine data from multiple sources and gather this information in order to achieve inferences*".

While a global definition has not yet been achieved, *data fusion*, as an overall term, can be explained in a broad sense as the result of combining data from several sources into one set of data that has higher quality than its parts.

5.3.2 Data Fusion classifications

Data fusion can be classified according to different features. In this section, a classification based on the relationship between input data coming from sensor nodes, and a classification based on the levels of abstraction of the data used in the fusion process will be presented. Additional classifications exist, such as a classification based on Input/output abstraction level of data [113], but will not be addressed as are not relevant for the protocol to be built.

Classification based on sensor relationship

By analysing the relationship between the input data coming from different sources, data fusion can be classified in *complementary*, *redundant* or *cooperative* (Fig. 5-2) [114].

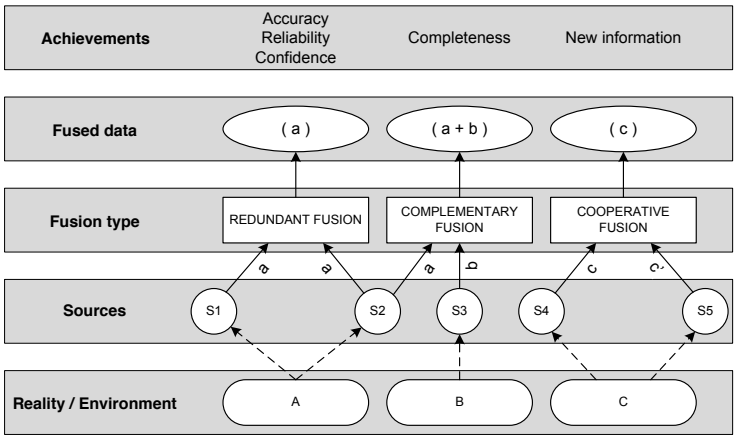


Fig. 5-2 - Information Fusion based on relationship between sources - adapted from [108]**,** [115]

Redundant fusion is when different sources provide the same data, that combined provides for a higher level of accuracy, together with increased levels of reliability and confidence. At the same time, in WSNs, fusing redundant data avoids the unnecessary transmission of data, saving valuable energy. This type of fusion is also known as *competitive fusion* [115] as each source competes for the right value, and a value that differs from the majority may be discarded by voting. Given sources *S1* and *S2*, providing the same data *a*, the fused information is a more accurate *a*, that can be denoted as *(a)*. An example of redundant fusion is the capture of a sound using different microphones that together reduce noise and provide for a more reliable and accurate sound capture.

In complementary fusion the different pieces of input data provided by sources present different aspects of reality, that combined provide a more complete understanding of the scenario. If the sources do not depend on each other, then the different pieces of data are not redundant but complementary, all contributing to a better knowledge of the environment being sensed. Given Sources *S2* and *S3*, providing data *a* and *b*, the fused information can be denoted as *(a+b)*. An example of complementary fusion is the combination of different video cameras and temperature sensors of a room to create one image in which the different temperatures are shown.

Cooperative fusion happens when different sources provide data that is combined to produce a new piece of data (usually more complex) that is not available from any single source. Given Sources *S4* and *S5*, providing data *c* and *c'*, the fused information can be denoted as *(c)*. An example of complementary fusion is target location where by combining different parameters like angle and distance, target localization can be established.

Classification based on levels of abstraction of the data fused

Three hierarchic levels are usually used to characterize the abstraction used by the data fusion process: *data*, *feature* and *decision* [113]. The corresponding data fusion processes are often referred by *low-level*, *medium-level*, and *high-level* fusion. Low-level fusion combines the different sources of raw data to produce a more accurate piece of data. Medium-level fusion combines attributes or features of the sensed entities (e.g. shape, edges, position, speed, temperature) to produce a feature map that can be used by other tasks. High-level fusion combines decisions made by processes that may include voting, fuzzy-logic, Bayesian approaches, any statistical process, among others. The resulting decision is expected to have improved confidence.

While the initial classifications did not include the possibility of a fusion process use, at the same time, different levels of abstraction, a new level was later added. Multi-level fusion is used to represent the combination of data from different levels (e.g. a measurement is fused with a feature to provide a decision). [108]

5.3.3 Data Fusion Architectures

The data fusion process can be generally divided in two separate categories: *centralized fusion* or *distributed fusion*. In a centralized fusion process raw data from different sources is sent through the network to a central point where it is processed altogether. In a distributed fusion process data is processed along the path, in intermediate nodes, or even in the same node that collects data. Either option may be adopted according to the specific needs of the application used. Next, some advantages and disadvantages of each option are presented.

Centralized data fusion (Fig. 5-3):

- Advantages: the fusion process has a global knowledge of the network;
- Disadvantages: data may not be available when needed, causes high overhead in network.

Distributed data fusion (Fig. 5-3):

- Advantages: the overhead is distributed along the network;
- Disadvantages: data fusion is incremental, not having a global view of all data, suffering several fusions along the way.

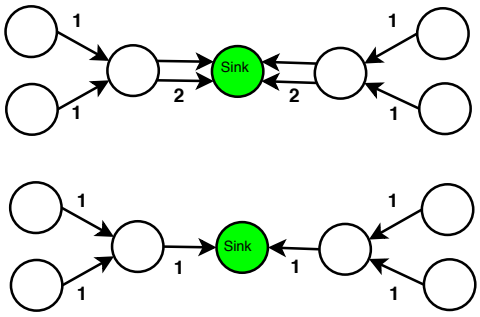


Fig. 5-3 - Centralized (above) versus distributed (below) data fusion. Arrows represent data packets sent.

5.3.4 Data Fusion techniques

The process to fuse data depends on the objectives desired. Many techniques, methods and algorithms have already been developed to reduce the amount of data exchanged, filter noise, predict and infer about a monitored entity. In [108] authors divide them in different classes according to their purpose: inference, estimation, classification, feature maps, reliable abstract sensors, aggregation and compression. The more significant are described next.

Inference methods are applied in decision fusion, when decisions are made using a sequence of assumed true propositions. Some classical methods are Bayesian inference and Dempser-Shafer inference. The first provides a formalism to merge data according to rules of the probability theory, combining different evidences using conditional probabilities whose uncertainty varies between 0 and 1, representing states of complete disbelief to complete belief. The estimations are updated as further evidence is acquired. The Dempser-Shafer inference extends and generalizes the Bayesian theory, combining evidence from different sources to arrive at a degree of belief that takes into account all the available evidence. Other techniques include Fuzzy-Logic, which deals with approximate reasoning, based on imprecise premises, Neural Networks, used in classification and recognition tasks, and learning by examples, Abductive Reasoning, a method that infers causes based on resulting evidences, and Semantic Information fusion, where nodes only exchange resulting semantic interpretations of raw data, not the data itself.

Estimation methods were inherited from control theory and include techniques such as Maximum Likelihood, Maximum a Posteriori, Least-Squares, Moving Average filter (used in digital signal processing for its simplicity and for reducing random white noise while providing an adequate response), Kalman filter, and Particle filter.

Feature maps include Occupancy Maps and Resource Activity Maps, where some features are selected and represented in maps for use of applications. These maps are created from previously fused and selected raw data.

Aggregation is implemented by using common summarization functions such as those used in query languages (*max*, *min*, *average*, etc.). It explores the synergies of data, removing redundancies and improving the energy efficiency of the network data delivery.

Compression in a data fusion sense does not make a plain compression of the data. It uses the synergies between sources, and the semantics of the data, to reduce the size of the data in a form that would not be possible if the data was from a single source.

From all the techniques presented, data aggregation is the more simple to in-network processing and so the more adequate to be used in a distributed WSN data fusion approach. Data fusion in WSN will be addressed next.

5.3.5 Data fusion in WSNs

A typical WSN deployment comprises several sensor nodes, each producing a large quantity of data that needs to be processed and delivered to the sink, according to application specifications. These networks are built to gather and process data from the environment, being it a constant flow of data or a special event that arises. In either case, data must be collected and transmitted to one or more sinks that depend on the information delivered by the all network.

5.3.5.1 Overview

Excluding scenarios where each sensor has specific and unique tasks, data generated from neighbouring sensors is often redundant and highly correlated. The natural redundancy present in most of WSN does not happen by chance. The fact is that several sensors placed in neighbouring places, not too far away, tend to catch the same phenomenon. This characteristic is often promoted by who deploys the network, trying to achieve robustness and fault-tolerance by adding redundancy. Also, the uncontrolled scenarios where WSN are typically deployed, cause many kinds of interference to the measurements taken by sensors (e.g.: variation of pressure, temperature, radiation, electromagnetic noise), resulting in errors and incorrect measurements, that add to the natural problems faced by low-cost WSN technologies – not very precise sensors, limited range and energy problems. Having redundancy is a way to minimize the resulting errors and the lack of accuracy, both by being able to use additional similar data to detect and remove abnormal values, and to combine several sources into one more reliable value.

In the case of controlled WSNs, there are strict performance and QoS goals that need to be assured. These networks are built to use the flexibility of traditional WSNs with critical applications such as those for industrial environments, health or military. In these networks the objective of data fusion comprises not only the fusion of the data sensed by the network but also the performance data used to monitor and control its performance. While for the first the objectives are the same as for generic traffic in traditional WSN,

for the second the objectives are to lower the overhead of measuring, saving and transmitting the performance traffic along the network, while providing for its accuracy.

In a conceptual view, data fusion may be viewed as having two parts. The first is the establishment of a network that will gather all the information needed. The second is the process of fusion of the gathered data itself.

5.3.5.2 Data gathering algorithms and protocols

WSNs have to collect data from all or part of the nodes constituting the network. The data collected is then sent to a base station commonly named Sink. Data gathering protocols configure the network and enable the collection of information from the nodes, considering data communication issues and data fusion needs. The architecture of the sensor network used plays a vital role in the performance of the different data gathering protocols and algorithms. Some network related issues of these algorithms will be now discussed.

Address versus data centric routing

Data fusion in WSNs is more concerned with data and less with addresses since most of the times the goal is not to find the short route between different nodes, but to find the routes from multiple sources to a single destination that provide for the best in-network data fusion [60]. In this context, two routing schemes can be considered: *address-centric routing* and *data-centric routing*.

Address-centric routing: End-to-end routing that delivers data from each source to the destination (sink) using the shortest path.

Data-centric routing: The routing delivers data from sources to a destination (sink), with the objective of being data efficient. Along the path, it performs a data analysis of the packets, aggregating data when possible.

The special characteristics of WSNs tend to favour data-centric routing protocols, instead of the typical address-centric routing protocols.

Data gathering protocols categorization

Assuming that the data can and will be fused, some algorithms were developed to construct the path to transmit data. The process to collect the data varies according to the structure of the network and to the specific protocol operation and data fusion to be accomplished. Many authors provide different classifications that differ according to the specific issue considered.

In [116] authors divide WSNs data gathering protocols in four categories: flat, hierarchical, location based routing and network flow/QoS. In flat networks, different nodes have the same responsibility. Requests are spread through the network and

transmitted back to the sink. To minimize energy consumption, nodes aggregate data during transmission. Normally this architecture increases the redundancy of information. In hierarchical networks, nodes play different roles, some sensing and communicating in a short range, while others, having high energy, process and send the data over longer distances. This architecture increases the scalability of the network, the lifetime and energy efficiency. The protocols that use hierarchical architectures usually have two phases, a first phase where the cluster-head nodes are selected and a second phase where data is sent. Location based protocols address sensors by their location, leading to the prevalence of communication between neighbours. This architecture is normally used in scenario with low or no mobility, as all the routing considers the static position of nodes. In Network flow/QoS networks, routing is set up as a network flow problem, having in consideration different QoS metrics. Sensor paths are obtained by balancing the metrics, to achieve low energy consumption and high data quality.

While discussing data aggregation techniques, in [117] authors classify aggregation protocols based on the main characteristics that influence their operation: *network architecture*, *network flow*, and *QoS awareness*.

The first group, network architecture is divided in *flat* and *hierarchical networks*, the second item being sub-divided in *cluster*, *chain*, *tree* and *grid based*. In a flat network all nodes have identical roles, while in a hierarchic network each node's role depends on its place in the hierarchy. In cluster networks, data is transmitted to a local cluster head, which fuses all the data and sends it to the sink. In a chain-based network, sensors only transmit to close neighbours. Each node fuses its own data together with the received data and sends the fused data to its own neighbour. Eventually a leader node (equivalent to a cluster head) transmits the data to the sink. In a tree-based network, nodes are organized as a tree and data fusion is done along the tree. Finally, in a grid-based network, the space covered by the network is divided in a grid. Each sensor belonging to a region transmits to the aggregator of that region, which in turn sends the fused data to the sink. Table 5-2 presents the main characteristics of flat and hierarchical networks.

TABLE 5-2 – FLAT AND HIERARCHICAL NETWORKS COMPARED (ADAPTED FROM [117])

Flat Networks	Hierarchical Networks
Data aggregation performed along the multi-hop path or just in the sink.	Data aggregation performed in cluster heads or in a leader node.
Data aggregation routes are created in regions that have data for transmission.	Overhead involved in cluster or chain formation along network.
If sink fails the all network fails.	Network does not depend on a particular cluster head.
Heavy communication needs as each node must transmit all data to the sink (in the case where aggregation is only done at the sink).	Less communication needs.
Higher latency as data must reach sink.	Lower latency as nodes send data to nearest cluster head.
Optimal routing can be guaranteed with additional overhead.	Routing structure is simple but not necessarily optimal.
Does not use node heterogeneity for improving energy efficiency.	Node heterogeneity can be exploited by assigning high-energy nodes as cluster heads.

The concept of network flow and QoS aware classification of aggregation protocols is also introduced, as these protocols cannot be characterized by the subjacent network but by the specific approach to the problem. These protocols treat the network as a graph and aggregation is addressed as a network flow problem. The special case of QoS aware protocols, target the guarantee of QoS metrics (such as bandwidth, end-to-end delay and information throughput) whose measurements are added to the network flow problem.

In [118] aggregation tree protocols are classified based on their specific tree structure: *planar tree*, *simple cluster* and *cluster-tree structure*. In planar tree structure algorithms, data is collected from children nodes and sent to their parent to be fused, which will then send it to their own parent. In cluster structure algorithms each node sends its information to a cluster head. Each cluster head fuses the data received and send it directly to the sink node. Cluster-tree structure algorithms mix the two previous types by having nodes send their information to a specific cluster head. The set of existing cluster-heads form a tree, where each cluster-head sends its fused data to its parent cluster-head.

Distributed Data Fusion approaches

The relation between the data fusion processes and the existing data network results in different approaches to the fusion process. As seen before, distributed data fusion, in opposition to a centralized approach, is the model that better fits the special characteristics of WSNs. In [108], authors present different distributed computing paradigms that may be adopted by WSNs: *in-network aggregation*, *client-server*, *active networks* and *mobile agents*. In in-network aggregation, the most used paradigm, data fusion is performed while data is on route to the sink. Depending in the network architecture and on the gathering protocol, the fusion process may take place in different nodes (e.g. all nodes, intermediate nodes, cluster-heads, etc.). Client-server approach applies typical client-server architecture, normally adopting a data centric approach. Information is sent to specific servers that fuse data in response to queries from the

clients. Active networks use programs injected in the network to perform the needed computations, in a highly flexible way. If the network conditions or objectives change, another program that responds to those questions is injected in the network providing the necessary answers. Finally, in mobile agents paradigm, programs travel the network performing data fusion.

5.3.5.3 Data fusion methods – special case of aggregation functions

As it was addressed earlier, aggregation is a part of the data fusion processes, the one that deals with summarization. The simplicity of these functions makes them very used in WSNs scenarios. The special case of data aggregation has already been presented in Section 4.4.2.

5.4 FUSION AND THE SPECIAL CASE OF WSN PERFORMANCE MONITORING

Having reviewed data fusion, the concepts learned and the specific characteristics of performance monitoring will be analysed to create the basis for a new performance data gathering protocol for WSNs.

First, in Table 5-3, the classifications reviewed in Section 5.3.5.2 are joined in a new general overall classification of data gathering protocols, in a WSN perspective, based on network organization.

Having in mind that the WSNs with more restrict QoSensing needs are carefully planned, that performance messages as extra traffic should be minimized (to minimize the overhead caused), that the existing overhead should be distributed to avoid depletion of specific nodes and that a network where all nodes have the same resources is a more generic assumption, the best choice is a decentralized hierarchical tree based gathering protocol. A centralized fusion would imply that all performance messages had to be distributed hop by hop, individually to the sink, spending extra energy and causing interferences along the path. A decentralized hierarchical chain based approach does not guarantee a time limit for the data to reach the sink and implies extra management in the establishment of the chain. A location based grid protocol relies on special nodes to fuse data, creating the need for special nodes with improved resources. Using a cluster-based approach implies extra maintenance in choosing the cluster heads and also implies extra effort at some specific nodes. A tree based protocol, while not immune to eventual maintenance operations is more simple, guarantees a specific number of hops to the sink and does not require the use of special nodes.

TABLE 5-3 – NEW CLASSIFICATION OF DATA GATHERING PROTOCOLS BASED ON NETWORK ORGANIZATION.

Centralized	In a centralized network each node sensed data is sent as it is obtained to a unique central fusion node. These networks may be multi-hop.		
Decentralized	In a decentralized network there is more than one fusion node.		
	Flat	In Flat networks every sensor node has the same role, sensing and treating the data. Data fusion is made along the path using data centric routing.	
	Hierarchical	Nodes are related hierarchically with fusion being made at each hierarchy level or just at special nodes.	
		Cluster based	Data is transmitted to a local cluster head, which fuses all the data and sends it to the sink.
		Chain based	Sensors only transmit to close neighbours. Each node fuses its own data together with the received data and sends the fused data to its own neighbour. Eventually a leader node (equivalent to a cluster head) transmits the data to the sink.
		Tree based	Nodes are organized as a tree and data fusion is done along the tree.
		Location based (e.g.Grid based)	The space covered by the network is divided by location (a grid in Grid based networks). Each sensor belonging to a region transmits to the aggregator of that region, which in turn sends the fused data to the sink.

The use and efficiency of data fusion depends on many items such as traffic characteristics, available hardware, network restrictions and goals to obtain from the fused data. WSNs new application domains require the control of specific levels of performance and QoSensing. This control can be made sporadically, in real-time, or by request, depending on the specific applications used, and performance objectives established. According to each scenario specific needs, the performance data to be transmitted can be fused or must be sent individually to the network performance control centre (connected to the sink). The control of performance can also be done differently depending on the network phase: *deployment*, *operation* or *debug*.

The use of performance monitoring in critical WSNs has some additional characteristics. These networks must provide QoSensing assurances, but that assurance must be done continuously, and, if possible, close to real-time. If, for some reason the existing QoSensing does not match the initial demands, the control centre must be informed and some intervention in the network may have to be done. Therefore, it is essential that these networks continuously measure and control the existing performance, and that information is always sent to a control centre connected to the sink. However, having a constant flow of performance information travelling in the network, pressures the performance of the network. Additional traffic, that adds up to the existing non-performance data, not only wastes nodes valuable energy, but also creates overhead in the communications, CPUs and memory of each node, decreasing the performance that was the primary goal to assure. The use of data fusion in order to reduce the performance

traffic is a powerful resource to minimize the overhead while providing for the necessary performance control. As these networks tend to be carefully planned their topology is normally known, and their delivery tree is mostly static. Also, in many cases it is possible to avoid procedures common to auto-configuring networks like the creation of delivery trees or the selection of cluster heads that aggregate data. In these cases in-network aggregates distributed in the network can be a more appropriate solution, lowering the complexity of the code in the nodes and increasing the lifetime of the network. Also, by using a delivery tree, with each node having a specific father, duplicate values are not a problem. In order to avoid unnecessary waste of energy, care must be taken in choosing simple calculations and low memory usage, when fusing data. Finally, the process must be feasible either in one-hop networks as in multi-hop ones.

As the values returned by nodes, either as result of any malfunction, or from a security attack, or caused by transitory problems that are part of the intrinsic nature of WSN, are outside our control, care must be taken in choosing the right method of data fusion. These malfunctions may affect one or many nodes. However, it is assumed that only a small percentage of all the nodes suffer from persistent malfunctions while a high percentage may suffer from transitory problems because they are being affected by the same exterior influence. In all cases, if the problem persists longer than a specific time frame, out of the predefined value range, the central control system must be notified.

To address the issue of performance monitoring, specific metrics must be considered, along with a new method to its calculation and delivery. In the previous chapter performance metrics were categorized as individual and collective. Individual metrics are metrics that relate to only one node and that are transmitted end-to-end. In collective metrics, the reality observed is not individual but involves part of the network, or even the all network. Collective metrics are the natural answer to the measurement of the global QoSensing of the network. However, if used alone, they would hide individual abnormal values. On the other hand, they could be biased from a specific individual value from a malfunction node or as a result of some security attack. To avoid both situations, metrics should be bounded. In case of an out-of-bound value is detected, an alert must be generated and sent directly to the sink. This specific value should not be included in the collective metric. Individual metrics can be always used when debug is necessary, being available through direct query.

Next table summarizes some of the differences between regular data traffic that results from collecting data from sensors, and specific performance data. Data monitoring in the sense of controlling sensed values (not just collect them), is in this context included in performance monitoring. This analysis is essential to be able to optimize a fusion process that serves the purpose of networks with controlled performance.

TABLE 5-4 – PERFORMANCE MONITORING VERSUS DATA COLLECTION, IN WSN.

Data Collection	Performance monitoring
Raw data sensed is subject to interference from the environment that result in abnormal errors, and affects the accuracy of the sensor devices.	Metrics are calculated based on data received or internally measured, less subject to variations due to environmental sources. However, some metrics may be influenced by exterior hazards (e.g delay, lost packets).
Raw data is subject to high levels of redundancy if different nodes measure the same events.	Redundancy is low.
Specific values of the data are required to assess the subject sensed (e.g. multiple different valve measurements cannot be fused).	Original performance data is not needed most of the times; what is needed is to guarantee that performance levels are guaranteed, and to have an idea of their evolution in real time. In case of need specific metrics should be queried.
Data transmission can be triggered in the occurrence of an event, being silent in the rest of time.	Nodes must provide, at constant time intervals, a proof that they are alive and with the expected performance levels.
Is the primary reason for the existence of the network, and therefore its traffic should have priority.	Its traffic controls the network performance and should not create overhead to existent traffic. The use of the existing traffic for carrying performance information, should be used whenever possible. Only alert messages have high priority.
Values are always fused or never fused.	Metrics may need to be sent fused or in individual values, depending on the requirements of the network at the time; both values may coexist in time. Also, some metrics may be fused for some nodes and not fused for others.
Different sensed values normally have the same fusion requirements.	Different performance metrics have different fusion requirements.

From the analysis it can be seen that while the redundancy of performance data is low, its specific values are not needed in most occasions, especially if considering that the performance of a WSN is subject to normal variations caused by the intrinsic nature of these networks. However, a constant flow of performance data, together with a global knowledge of the performance being provided, is needed to assure that the network is behaving correctly, even if there is no actual data sensed to report. Furthermore, if using data fusion techniques on performance data, it should not be assumed that the same technique would be used for all types of performance data.

This analysis will be used in the design of a new protocol for performance collection.

5.5 A PROTOCOL FOR WSN PERFORMANCE COLLECTION USING DATA FUSION

This section describes a new protocol – *WSNs Metrics Collection and Alert Protocol* (WMCAP) - to be used for performance collection in WSNs. Although designed to the specifics of performance data monitoring demands, the protocol can also be used to deliver the sensed data itself.

5.5.1 Motivation for a new protocol

The use of WSNs in performance controlled scenarios implies not only that the network must assure a predetermined QoSensing level, to provide for application specific demands, but also that the QoSensing must be maintained over time. The latter implies the use of performance monitoring data that is continuously sent from the nodes to a base station, which, in WSNs scenarios, presents some new problems. The typical implementation of network monitoring involves the collection of performance data from each node to a central unit. As data is (most of the times) delivered in a multi-hop fashion, each node being responsible for the transmission of all the data from their children, that creates huge amounts of data that must be retransmitted through the network, wasting not only valuable energy but also creating substantial overhead over the wireless medium. At the same time, the network overhead introduced by the monitoring will, by itself, degrade the performance of the network, as regular data will have more difficulty to be transmitted.

In spite of not being available specifically for performance data, several data fusion protocols exist for normal sensed data. The use of these protocols to diminish the amount of performance data poses two problems. First, performance data would have to be subject to the same rules of other data, preventing the use of alerts and extending the packet sizes. Secondly, performance data would be subject to the data timing of each node, or from the data delivery model used, not being able to give a real time indication of the network performance. On the other hand, the use of a fusion protocol adapted to performance metrics diminishes the need from a constant data feedback from nodes, which may adopt an event-driven data delivery model or at least lower the rate at which are sending data.

In WSNs with performance control, specifically in industrial environments, the redundancy of data is lower or non-existent. The data transmitted from nodes may not be a target for fusion as the data may refer to different realities. As an example, a WSN may be monitoring a set of valves, but each valve may have different appropriate values that relate to the specific substance it addresses. Fusing these values is not an option, and sending performance metrics with them is not efficient, especially because not all nodes sense data but all nodes produce performance metrics. Also, these WSNs tend to have static topologies that use almost static delivery trees, with changes being the exception.

However, if data being sensed has similar needs to performance data, namely the same cadence and the possibility of being aggregated or sent as an alert if out of specific boundaries, data and performance data may be sent together using the same protocol.

In order to deal with the specificities of performance collection and monitoring in a WSN scenario, a new protocol is now proposed. This protocol merges different techniques, from inferred metrics to the use of data fusion and collective metrics.

5.5.2 Protocol goal and requirements

The goal of the protocol to be proposed is to enable the collection of performance data from the nodes to a sink, with low overhead, while maintaining the possibility of an effective monitoring of the WSN global performance, and allow the immediate report of any malfunction.

The protocol will work in the application layer, and work with any existing routing protocol. To do the performance collection, every node will periodically calculate its metrics and send them to its upstream neighbour. The metrics to be sent, their cadence and healthy intervals can be set by the protocol during operation or set during the deployment of the WSN. The effective method to calculate the metric is beyond the protocol and must be known by the node. In order to lower the overhead in the network, each node fuses its own metrics with the metrics received from their children, producing collective metrics that will be sent hop-by-hop in consecutive fusions until reaching the sink. Sending individual metrics is also possible but should be an exception. However, only using collective metrics that are sent hop-by-hop, that have to be fused along in each hop, and that are only sent in a pre-defined cadence from each of the nodes, causes some problems. The first problem is that it is difficult to detect any malfunction in the network. Collective metrics, as explained in Chapter 4, may hide a specific value as it is fused along with many others. Also, even if the malfunction can be spotted from the collective metric, the time between the calculation of the problematic metric and its reception in the sink may exceed the required interval. To solve the mentioned problems alert messages were created. Whenever a node detects a metric out of healthy bounds, it immediately sends that metric to the sink, without being fused, using an individual metric in a special alert packet. The same happens if a node does not hear from its children for a specified time. The protocol also provides simple debug functionalities, allowing the query of specific metrics from the nodes. More details about the mechanisms used by the protocol are addressed in following sections. A detailed explanation is given in Appendix A.

The new protocol intends to fulfil the following requirements:

- Enable a performance monitoring that minimizes the impact in the network lifetime, especially considering the stress on the resources of nodes as their position raises in the topology, due to the increased number of messages to forward;
- Allow for the immediate report of alerts when any malfunction in the network is detected;
- Disregard sporadic losses of performance due to transitory events;
- Provide for simple performance debug functionalities.

5.5.3 Protocol details

After determining the requirements, the details behind the new protocol are presented.

Initial assessments

In the WSN every node may have a sensor(s) and every node must be monitored. Also, intermediate nodes are responsible for routing data and may be responsible for its treatment and preliminary analysis. In face of this, the proposed protocol will:

- Use a data fusion mechanism to lower the data overhead in the network;
- Use collective metrics to report the global QoSensing of the network;
- Allow changing of node metrics thresholds during deployment and operation;
- Deliver individual metric values when previously defined thresholds are crossed – in the form of an alert;
- Enable query of specific metrics to each individual node (debug mode);
- Allow the specification of the cadence of performance metrics delivery to each individual WSN specific needs.

The protocol will also use a MIB to save both local parameters and metrics received from the node's children. MIB in this context is used generically as a management table not directly correlated to the MIB used by SNMP.

Metrics treatment

There are 4 groups of metrics that must have different treatments:

- Metrics that can be inferred from normal data traffic (e.g. packet delay, number of hops, packet loss - if a sequence number is present);
- Metrics that must be explicitly sent timely (e.g. energy levels, total packets sent);
- Event-driven metrics resulting from the crossing of a predefined threshold (e.g. a metric measured in nodes whose value is sent as an alarm);
- Metrics sent as a response to specific queries (e.g. a specific metric needed by the user or controller process).

The first group of metrics does not have a special network treatment and results from reading the data traffic reaching the sink. Care must be taken if data packets are themselves subject to any aggregation or fusion process, what might alter the performance data inferred from packets, resulting in erroneous values (e.g. original sent timestamps may be altered) or in the loss of valuable data (e.g. specific fields like timestamps, or number of hops may be removed).

The second group is sent through the network in specific performance packets. These packets will be treated by the protocol and be subject to a specific treatment in order to lower the overhead caused in the network. This group will use mainly collective metrics, if specifically required individual metrics can also be used.

The third group results of in-node evaluation of metric values. When a metric has a value out of the expected interval no data fusion is performed and the metric is sent directly to the sink as an alert.

The fourth group of metrics is sent in specific performance metrics packets as an answer to specific queries sent by the sink node. These queries will be used when the network is in debug mode, after a problem is detected. The queries can be sent to individual nodes or to all nodes.

Architecture

The protocol is conceived to work with WSNs with strong performance monitoring needs, which have strict QoSensing targets to comply. Normally these networks are subject to careful planning and design as to potentiate their performance capabilities, not implementing auto-organizing protocols to establish the network. The protocol performance also benefits from having multi-hop networks, with cluster based networks and hierarchical tree networks being the target scenarios. The protocol will adopt a distributed fusion mechanism, with fusion happening in all or in some specific nodes in order to minimize the number of packets in the network.

Node types

There are three types of nodes, by functionality, used by the protocol:

- Simple sensor nodes;
- Sensor nodes with data treatment roles (referred hereafter as common nodes);
- Sink node;

Simple sensor nodes are nodes that do not perform any data treatment. They just communicate their own data or forward received data. In the case of leaf nodes, the only action performed is the sending of data to intermediate nodes in multi-hop networks or to the sink in one-hop networks. Intermediate nodes belonging to a multi-hop network also forward packets received from neighbours.

Sensor nodes with data treatment roles participate in the treatment of their own data and in data coming from other nodes. These nodes will have the overhead of saving more than their individual data and to manage, perform calculations and transmit data that includes values received from other nodes.

The sink node (multiple sinks are also possible) receives data from the network, both individual as fused, and may perform some additional processing to the data. The sink, contrary to other nodes, is assumed to have no restrictions in terms of energy, memory or computation power.

Data Fusion considerations

After having analysed, in previous sections, the fusion methods and their specificities when used in WSNs, the fusion method to be used by the protocol is now stated.

The protocol will use collective metrics, to assess the global performance of the network, calculated using a distributed data fusion mechanism. To produce those metrics, each node fuses its own metrics with the metrics received from its children. When computing collective metrics from the network, single out of bounds values must not significantly affect the result. Otherwise, just a single wrong value could affect the overall calculated network performance, not giving a true view of the global performance. For example, a scenario can be imagined where packet loss is normally very low. However, if someone takes an electronic device that interferes with the transmission of a specific node, being that action intentional or not, the loss of packets sent from that specific node will rise substantially. If that metric was fused with metrics from other nodes, the resulting global performance would have a value that did not correspond to the actual global performance of the network, but only to the malfunction of a limited part. As it is clear from the example, the problem does not depend on the protocol used to transmit information, but in how the fusion process is made, reason why the focus of the protocol to be presented is in the data fusion mechanisms.

While the protocol can use any type of fusion (the method of calculation does not invalidate the protocol mechanisms), for simplicity of calculation, aggregation functions will be the main method to fuse performance data. To enable it, the functions used in the fusion should be decomposable. A decomposable function has the same value if calculated with all data or from the application of the function to different subsets of the same data. Function *f* is decomposable by a function *g* if it can be expressed as [97]:

$$f(a_1,\ldots,a_n)=g(f(a_1,\ldots,a_k),f(a_{k+1},\ldots,a_n)) \quad (5\text{-}1)$$

Functions such as *max* and *min* are decomposable. So is *average* if the number of elements in each computed average is known. On the other hand, the *median* can only be precisely calculated if all individual values are known to the function at calculation time. Non-decomposable functions may be used to fuse data that is sent directly to the sink, but are not suitable for consecutive in-network aggregation made in the path to the sink, unless less precision is acceptable.

However, as referred before (Section 4.4.2) *median* is more secure than functions such as *max*, *min* or *average* (the examples used). The reason is that an abnormal value does not invalidate the *median* aggregate. On the other hand, if an *average* is used, a single malfunctioning sensor returning random results can skew the average by an unbounded amount. To be able to use decomposable functions and prevent the effect of abnormal values, each specific metric that is aggregated should have specific healthy bounds. These bounds limit the permissible values of the metric and avoid exceptional cases (being the

result of the metric a real occurrence or the effect of an attack to the network). Therefore, out-of-bound values should not be included in aggregates but sent in separate packets as alerts, making the set more reliable. To deal with permanent problems of a node, it should be possible to disconnect a node, after it has positively been identified as being out of order.

Fusion can be performed in all nodes or in selected nodes, depending on the nodes where the protocol is active for a specific metric. This is controlled during deployment and after by the central station (using control packets sent from the sink). In all cases there is no need to select cluster-heads or specific nodes to make fusion, reducing the maintenance overhead in the network. Fusion is always done where available, and always in the path to the sink.

5.5.4 Protocol operation

The protocol operation will now be presented, including its messages and information needed in nodes memory. Additional information, flowcharts and details of the mechanisms and messages used are presented in Appendix A.

Setup

The initialization specifies which metrics the nodes should report and how, each metric's healthy range, the cadence of the reporting and a warning threshold for each metric.

The specification of the metrics consists in sending each node the identification of the metrics to calculate. The node must recognize each identification code and must also know the specific fusion mechanism or function to apply to that metric (this information must be programmed in each node). Finally, fusion settings of each metrics are also initialized. The fusion settings define if a specific metric is to be fused or to send directly and individually to the sink. The option for sending individual metrics should only be used in mandatory situations to avoid increasing the overhead over the network.

The metrics healthy range contains the acceptable values that the metric may have and translate the performance requirements of a specific WSN or application. The definition of the healthy range may be done automatically or explicitly. Specifics on how to determinate healthy ranges are outside of the scope of the protocol and depend on the specific WSN applications used and existing environment. If done automatically, the sink sends all nodes a message that requires the value of specific metrics. For this method to be successful, the network should be in a state that mimics the best-expected operation conditions. This operation is done preferentially during the initial network deployment phase. After collecting all responses from nodes, the sink defines (with a predefined margin), the acceptable ranges for each metric, based on the values obtained. This method does not impose the values to a network; it defines healthy ranges taking into consideration the existing WSN characteristics. If the definition of healthy values is done

explicitly, specific values are sent by the network administrator to all nodes, based on the requirements of the application to run.

Also, during the initialization, the reporting interval is set. The reporting interval is the time between performance updates, which are sent by the nodes to the sink.

The number of warnings tolerated for each metric can also be specified. The warning definition sets the number of times a metric may be out of the healthy interval without an alarm being sent. This is used to accommodate one-time fluctuations of metric values that represent transitory situations.

While the initial setup is done at network setup, it can also be changed dynamically during the network operation (in this case the explicit method for defining metrics should be used, as the network may no longer be under normal conditions).

After setup the protocol is ready to begin the performance retrieval. The algorithm of the setup process is shown in Fig. 5-4.

```
// Setup of the network by the sink
1 procedure setup_nodes()
2       if setup_mode == automatic
3               // send METRIC_QUERY to all nodes to get specific metric values
4               send_METRIC_QUERY_to_all_nodes()
5               // wait for responses from nodes (METRIC_QUERY_REPORT messages)
6               wait_for_METRIC_QUERY_REPORTs()
7               // Find suitable thresholds for metrics based on received values
8               calculate_thresholds()
9       else
10              // get explicit threshold values from the user
11              request_user_metric_thresholds()
12      end if
13      // sends child a SET_NODES message with threshold definitions
14      send_SET_NODES()
15 end procedure
```

Fig. 5-4 – WMCAP: initialization.

Operation

Once in operation, each node using the proposed protocol may receive data from upstream nodes (nodes closer to the sink) and from downstream nodes (nodes farther from the sink) (Fig. 5-5).

```
1   procedure common_node_main()
2       start_node()
3       while(TRUE)
4               // wait for new packet to arrive
5               packet_received == wait_for_new_packet()
6               // analyze source of received packet
7               if packet_received.source in {child nodes}
8                       process_packet_from_child(packet_received)
9               else
10                      process_packet_from_parent(packet_received)
11              end if
12      end
13 end procedure
```

Fig. 5-5 – WMCAP: reception of performance messages in a common node (1, 2 and 3 continue in Fig. 5-6 and Fig. 5-7).

Upstream nodes deliver setup messages and topology change messages to their downstream neighbours, as well as requests for specific sink metric queries. On receiving a setup or topology change message from an upstream node, each node modifies its own MIB to reflect the changes (MIB in this context is used generically as a management table). On receiving a specific metric request, the node calculates the requested metric and immediately deliveries it to the sink (Fig. 5-6).

```
1   procedure process_packet_from_parent(packet_received)
2       if packet_received.destination == this_node
           or packet_received.destination == all_nodes
3           if packet_received.packet_type == METRIC_QUERY
4                   // read metrics requested and send them to sink
5                   read_metrics_from_MIB()
6                   send_METRIC_QUERY_REPORT_to_sink()
7           else
8                   // update MIB with new configurations received
9                   update_MIB(packet_received.data)
10          end if
11          if packet_received.destination == all_nodes
12                  forward_received_packet_to_children(packet_received)
13          end if
14      else // packet is not for this node, just forward
15          forward_received_packet_to_children(packet_received)
16      end if
17  end procedure
```

Fig. 5-6 – WMCAP: reception of packets from upstream nodes.

A node may receive different messages from downstream nodes. If the message source is one of its 1-hop neighbours, it saves the time of the packet reception. This information is used to control the last contact from its 1-hop neighbours, as each node is responsible for the detection of the death or malfunction of its direct neighbours. From all the messages received from downstream neighbours, only metric report messages of fused metrics (*METRIC_REPORT_FUSION* packets) are treated. All other messages are just forwarded. On receiving a metric report message with fused data, the node reads its contents and saves them in its MIB (Fig. 5-7).

```
1   procedure process_packet_from_child(packet_received)
2       if packet_received.packet_source in {one-hop_neighbours}
3           update_MIB_with_last_contact_date(packet_received.source)
4       end if
5       if packet_received.packet_type == METRIC_REPORT_FUSION
6           update_MIB_with_metrics_data(packet_received.data)
7       else
8           forward_received_packet_to_sink(packet_received)
9       end if
10  end procedure
```

Fig. 5-7 – WMCAP: reception of packets from downstream nodes.

At regular intervals, defined in the setup, each node calculates its own metrics and validates if they are within the expected healthy interval (Fig. 5-8). If out-of-bounds metrics are found (and if the warning threshold was surpassed) an alert is generated and immediately send to the sink. Also, the time of last contact of all 1-hop neighbours is checked. If the last contact surpasses the defined threshold, an alert message is generated and immediately sent to the sink. Next, the node sends the individual metrics specified during setup to the sink. Finally, it reads the values of metrics to be fused that were

received from other nodes, fuses them with its own metrics and delivers a *MERIC_REPORT_FUSION* packet to its upstream neighbour.

```
1   procedure metrics_update_report()
2       while(TRUE)
3               // wait for update time
4               wait(UPDATE_TIME)
5               // calculate and verify individual metrics
6               for i = first_metric to last_metric
7                       value = calculate_individual_metric(i)
8                       if limits_crossed(i)
9                               send_ALERT_to_sink()
10                      else
11                              update_MIB_with_metric_data(i,value)
12                      end if
13              Next i
14              // Calculate and send fused metrics to upper neighbour
15              Calculate_fused_metrics()
16              send_METRIC_REPORT_FUSION()
17      end
18  end procedure
```

Fig. 5-8 – WMCAP: metrics update report.

The protocol uses four different fields for each fused metric. The first is its identification. The second is its value. Depending on the specific metric to address, the calculation that produces the fused value may vary. Furthermore, if needed, the calculation may be produced over one or more samples from each node, allowing for fusion in time and space. The third field is the number of samples that produced the resulting value of the fusion. The number of samples is important to allow the network manager and/or the monitoring software to know how many node values constitute the fused value. Also, it allows the network manager to detect if there are consecutive nodes failing to report, what may trigger a debug in the network. The last field is the number of fusion operations made to the samples, which indicates the number of data processing nodes that participated in the calculation of the final value. At each fusion enabled hop (i.e. at each hop that reaches a node where fusion operations are enabled), a metric report packet containing fused metrics is read and its information saved in the node's MIB. Then, at a specific cadence, the node fuses the values received from all neighbours with its own values and a fused value is produced. The number of samples that existed in each of the previously received packets is added with the number of samples that the node itself produced. The number of fusion operations is also added and increased by 1 (to add the present fusion). After all metrics are fused, the new packet containing the fused values of that sub-network is sent.

Messages used and internal information

The messages used by the protocol are presented in Table 5-5.

TABLE 5-5 – WMCAP: MESSAGES.

Sent by the sink	SET_NODES	Sets the metrics to be sent by nodes and specifies their healthy boundaries.
	METRIC_QUERY	Queries a specific metric from one or all nodes.
	ACTIVATE_NODE, IGNORE_NODE, REMOVE_NODE	Tells nodes to activate, ignore or remove other nodes from their neighbour list. If performance messages are sent by ignored or removed nodes, their upstream neighbours do not do not forward or process them.
Sent by common nodes	METRIC_QUERY_REPORT	Responds to a specific metric query made by the sink. Is never fused.
	METRIC_REPORT_SINGLE	Reports individual metrics from one node. Is never fused.
	METRIC_REPORT_FUSION	Reports a metric that has been fused with values from other nodes.
	METRIC_REPORT_FUSION_ RULE	Used in the extended version of the protocol, reports a fused metric defined by a specific fusion rule.
	ALERT_METRIC_REPORT_ OUT_OF_BOUNDS	Informs the sink that an out-of-bounds metric value has been detected in a specific node. Message is immediately forwarded to the sink.
	ALERT_DISCONNECTION	Informs sink that a specific node did not respond or sent a message in the predefined time limit. All nodes in the path to the sink immediately forward the message.
	ALERT_NEW_NODE	Message that informs the sink that there is a new node in the network. Used when mobility is supported.

In order for the protocol to work nodes must save some information in their internal memory. The information needed includes the metrics to report (specifying which to be fused), the information received in the *SET_NODES* message (healthy metric bounds, metrics update interval and alive threshold to detect the death of a neighbour node), a list of 1-hop neighbours with their data values to fuse and last time of contact.

Extended version

An extended version of the protocol is also proposed to address specific needs, such as the demand for the fusion of specific metrics in the nodes. This is achieved by using specific fusion rules. More details are covered in Appendix A.

5.5.5 Example of protocol operation

A simple theoretical example of the protocol operation will now be provided. In this example, the following conditions are assumed:

- All outliers are sent immediately;
- 1 metric will be used: node's energy;
- Values of energy by node: N2=100, N3=30, N4=90, N5=100, N6=150, N7=200, N8=40
- Minimum energy value per node = 50;
- Network errors are not considered.
- Fusion is accomplished by using the *sum* function;

The topology used and the number of packets that each network link transfers in one cycle of metrics reporting is showed in Fig. 5-9.

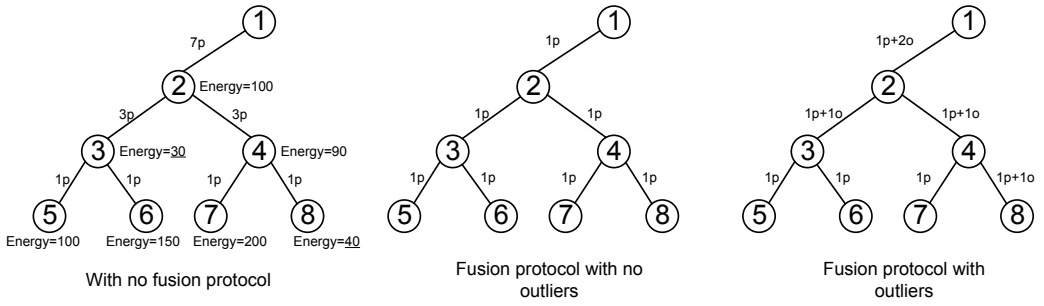


Fig. 5-9 – WMCAP example

In this example, only a metric is used. That metric is the node's energy, whose value is presented in Fig. 5-9 using the tag "Energy", next to each node. The letter 'p' indicates a message containing a metric value sent between two nodes. In scenarios where the protocol is active it corresponds to a METRIC_REPORT_FUSION packet. Letter 'o' indicates an ALERT_METRIC_REPORT_OUT_OF_BOUNDS packet that indicates that a metric value is outside a healthy range.

Three scenarios are presented. The first does not use any fusion protocol. In this scenario, the value of each node's energy is sent in a specific message, what translates in 1 message per link in the lower tree level, 3 messages per link in the second level and in 7 messages in the link to the sink, for a total of 17 messages in the network. The second scenario uses WMCAP but no outlier alerts are provided. In this situation in each reporting cycle only one message is sent per network link, for a total of 7 messages in the network. By fusing all the data from the nodes, and by knowing the number of samples fused a collective average of all nodes could be calculated in the sink, giving a global overview about the network energy. However, the information about the unhealthy metric values of nodes 3 and 8 is lost in the resulting fused metric passed along the network. With an average energy in the network of around 101.4, a value above the lower limit, the problems in the network are not detected. In the last scenario, WMCAP is used with alerts. This change adds the number of messages in the network to 12 messages but allows for the immediate detection of the problems, while maintaining the knowledge of the overall value of the network energy. Outliers are not included in the fused metrics.

Next, a message sequence corresponding to the setting of the example network and to the first cycle of reporting is shown.

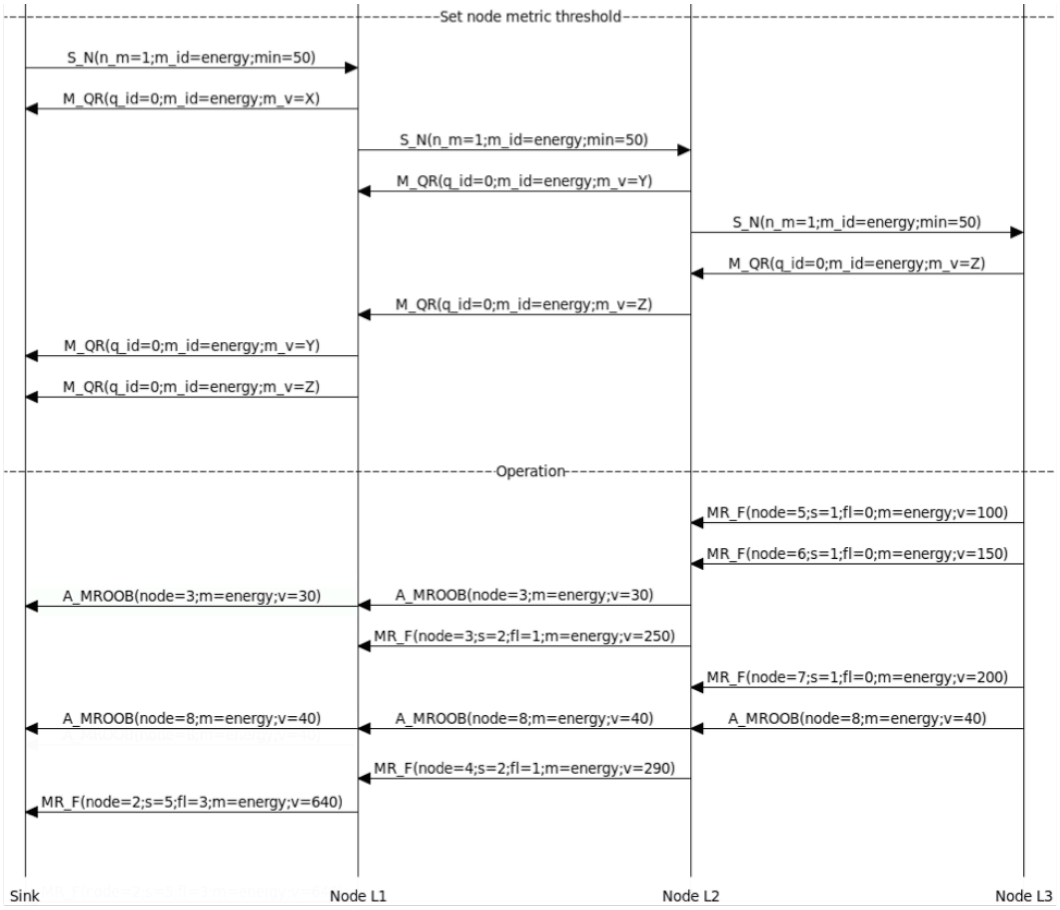


Fig. 5-10 – Sequence of messages used by the scenario using fused values and alerts

In Fig. 5-10, the values inside each of the messages correspond to the fields of those messages (see Appendix A). *METRIC_QUERY_REPORT* (M_QR) messages sent by each of the nodes, confirm the reception of the *SET_NODES* (S_N) message. Alerts are generated in the nodes where the outliers are detected and are immediately sent to the sink (A_MROOB messages). Fusion levels and number of samples, included in *METRIC_REPORT_FUSION* (MR_F) messages, are respectively the sum of all the fusions made to the metrics considered in the message and the number of the samples used to make the fusion. The value of the metric "Energy" is sent as the sum of the energy values of all the samples within healthy values (the average can be calculated by using the energy sum and the number of samples).

5.5.6 Scalability

To understand how scalable the protocol is, a tree topology in which every node (except leaves) has the same number of children, and where all branches have the same height (perfect tree), was considered. The number of performance packets generated in the network, in each monitoring cycle, was counted (using formulas in Appendix Section A.8) for a scenario where all nodes send performance packets and for a scenario where only leaf nodes send performance packets. The results are depicted in Fig. 5-11. The lines

represent the number of packets generated when using fusion (in both scenarios the number of packets generated is the same) and when no fusion is used. The columns represent the number on nodes in the network.

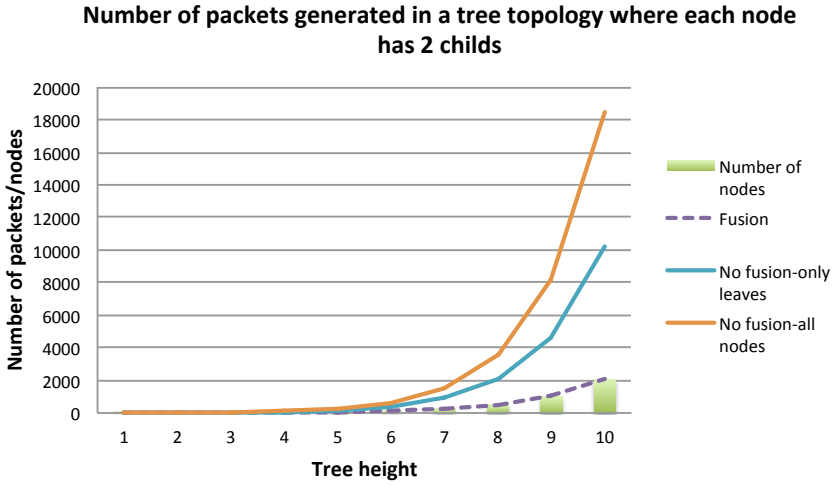


Fig. 5-11 – Number of packets generated in a perfect tree topology where each node has 2 children.

As can be seen, the fusion line follows the number of nodes in the network. In both alternatives without fusion the number of packets generated in the network rises fast as the height of the tree is incremented. A higher number of packets in the network causes more interference and depletes node's energy. Also, a rising number of packets is more problematic as approaching the sink, as a bigger number of packets may have to be forwarded. Using the same scenarios, the number of packets received by the sink in each monitoring cycle equals the number of sink's children when fusion is used (always 2 packets), and equals the number of nodes that send performance packets in the network for non-fusion scenarios.

By using data fusion in the network, the number of packets generated remains stable when considering the packets sent and received by each node. The total number increases as the network itself increases, being equal to the number of nodes used to send performance data.

In these calculations the number of alerts generated, that represent problems in the network, was not considered.

5.6 EVALUATION OF THE PROTOCOL

In order to assess the capabilities of the proposed protocol and to compare it to other approaches for performance data collection, simulation tests were accomplished.

5.6.1 Topology

The topology used consisted in a network of 32 nodes using a hierarchical tree. The topology and existing routing are depicted next (Fig. 5-12). The circle depicted is the transmission range of the sink, i.e., the distance within which all packets sent by the sink are received by a destination node (except if interference exists). All nodes have similar transmission ranges.

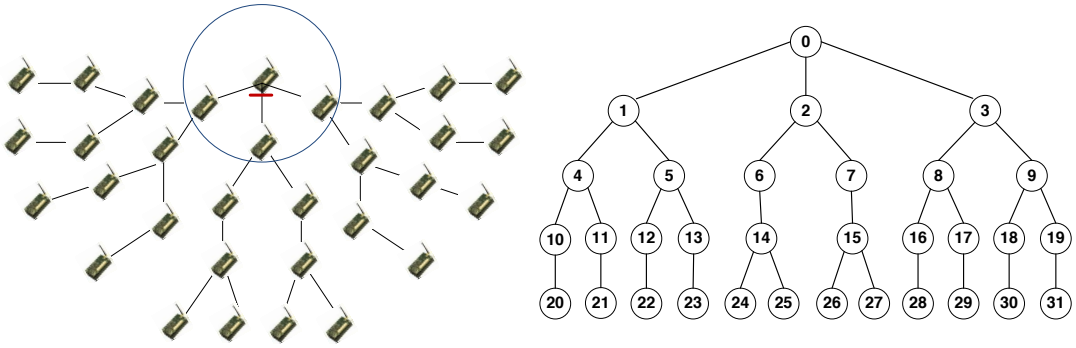


Fig. 5-12 – Evaluation topology - physical (left) and logical (right).

5.6.2 Simulation details

The simulation was done using the Cooja simulator [89] emulating 32 Tmote Sky nodes, each with a transmission range set to 40 m. The area simulated was of 240x120 m. All nodes report to the same sink (node 0) hierarchically. Send and receive ratio was set to 100% in the simulation setup file, which means that nodes will try to deliver 100% of the packets, no error is inserted in the links. The implementation of the radio uses *Cooja Unit Disk Graph Medium* (UDGM) [89] and is more restrictive than real radio communications. While in a real scenario interference may allow for the reception of a packet, in Cooja, overlapping packets always cause a packet drop, what potentiates the number of losses in networks with congestion. The network stack implemented in ContikiOS has 3 layers between the Network and Physical layers. These 3 layers are the MAC, *Radio Duty-Cycle* (RDC) and Framer. For the purpose of this simulation *Carrier Sense Multiple Access* (CSMA) protocol was used in MAC layer, which allows tentative retransmission of a packet if collision is detected. In the RDC layer, ContikiMAC RDC Protocol [90] was used. It was put in all nodes, including the sink (that normally is connected to power and because of that does not need to save energy). The reason to include the protocol in the sink regarded its possible use as a wireless gateway to a central station. ContikiMAC is a radio duty cycling protocol that proposes some enhancement over X-MAC, and that uses periodical wake-ups to listen for packets from neighbours, while remaining with radio off the rest of the time to save energy. The Framer layer uses "Framer-802154" a ContikiOS implementation compatible with standard IEEE 802.15.4 (2003). The ContikiOS Rime stack [24], which is a set of custom lightweight networking

protocols designed specifically for low-power wireless networks is also used. The transmission used in not reliable.

The use of an energy-saving RDC also increases the congestion and leads to a high loss of packets. When a collision is detected, the link layer (using ContikiOS CSMA protocol) tries to retransmits packets 2 more times. After that, the packet is discarded. As nor the data feed nor the performance feed are reliable, there is no guarantee that the messages arrive to destination.

In the simulation, regular data packets (that simulate the delivery of sensed data) contain 12 bytes of data. The *METRIC_REPORT_FUSION* packets, which contain the performance metrics, carry, for simulation purposes, 3 metrics: energy, delay per-hop and number of data packets already sent. Energy spent was calculated by a software-based power profiling mechanism of Contiki [23]. The mechanism runs directly on the sensor nodes and provides real-time estimates of the current energy consumption [87].

5.6.3 Scenarios and performance collection strategies

In order to test and compare the protocol, two scenarios were used. Next, each of the scenarios is described.

Scenario 1 - In this scenario, performance packets and regular data packets (that simulate sensed data) are generated and sent by all nodes in the network (with the exception of the sink) with a cadence of one performance packet and one data packet generated and sent every 6 seconds. This causes a high level of congestion in the network as data messages are all being sent through the intermediate nodes to the sink.

Scenario 2 - In this scenario the network has less congestion. Only leaf nodes create regular data packets, at a cadence of one every 9 seconds. Performance packets are sent by all nodes at a cadence of one packet each 6 seconds. As less packets travel through the network, the interferences are low.

Scenario 3 - In this scenario the network has less congestion due to the fact that there is no regular data transmission. Performance packets are sent by all nodes at a cadence of one packet each 6 seconds. As less packets travel through the network, the interferences are low.

In order to make the collection of performance data from nodes, 4 different strategies were used in the tests for comparison.

Strategy 1 (*Metrics fused + data*) – Uses WMCAP for performance collection and also includes a constant feed of regular data from all nodes in the topology.

Strategy 2 (*Metrics fwd + data***)** – Performance collection is done using dedicated individual packets and there is also a constant feed of data from all nodes in the topology. Each of these packets is just forwarded to the sink, with no processing done in any of the intermediate nodes. Performance packets and data packets are generated and sent.

Strategy 3 (*Metrics & data fused*) – Performance collection and regular data transmission are done using the same packets. Both performance and regular data are fused along the path to the sink. Packets containing both performance and sensed data are generated and sent. This strategy only applies when data can be fused what may not be always the case in real world practice.

Strategy 4 (*Metrics & data fwd*) – Performance collection is transmitted using the same packet as regular data but fusion is not used along the path to the sink. Packets containing both performance and sensed data are generated and sent. This strategy may be used when all nodes send sensed data.

5.6.4 Results

Having in consideration the scenarios and strategies defined previously, simulations were designed and implemented. The first tests compare the four strategies under Scenario 1. A comparison is made first in terms of energy spent and then in the number (and type) of packets generated. 70 cycles of performance reporting were simulated in each test made, and an average of the results is presented. Reliability is also addressed. The tests were repeated a numerous of times and their averages are presented next.

Next figures show how the protocol compares in terms of energy spent in the network and per each selected node (nodes from different levels of the network tree were chosen - Fig. 5-12).

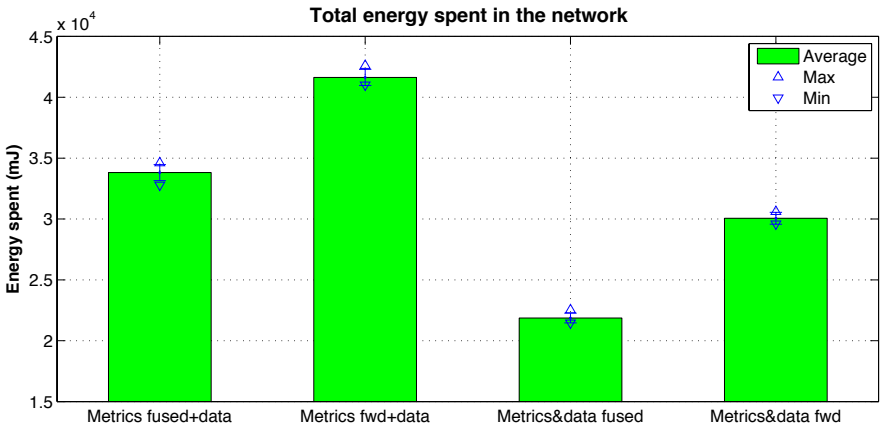


Fig. 5-13 – Scenario 1: Total energy spent in the network by strategy used (average).

The average total of energy spent during the simulation time is showed in Fig. 5-13, together with the maximum, minimum and standard deviation of the set of tests made. As

can be seen, the approach with better results is Strategy 3, fusion of both performance and regular data. The worst option is to use Strategy 2, which sends performance and regular data in different packets, both not fused along the network. Next, energy spent will be analysed by node (Fig. 5-14). Nodes 1, 2 and 3 are 1-hop away from the sink, nodes 4 and 7 are 2-hops away, nodes 12 and 15 are 3-hops away and finally nodes 23 and 25 are 4-hops away.

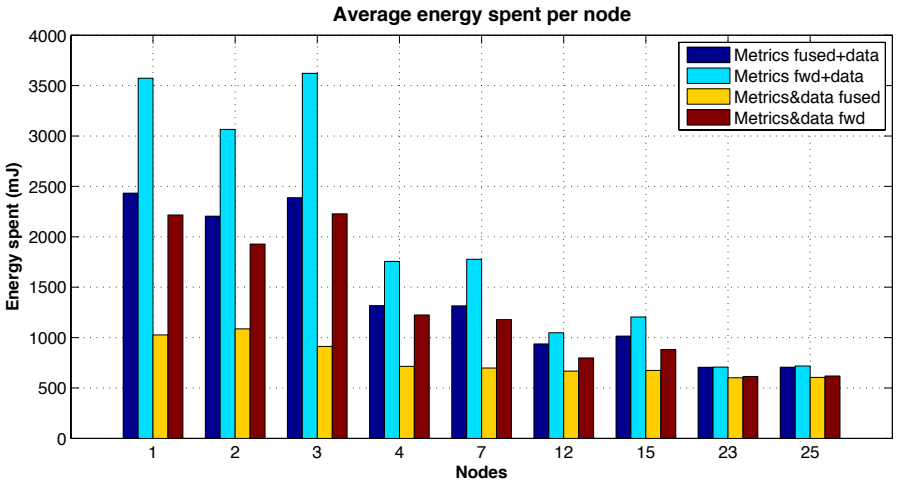


Fig. 5-14 – Scenario 1: Average energy spent per node by strategy used.

As it was expected, the energy spent is higher as the node gets closer to the sink. This tendency happens independently of the strategy used. However, as nodes get closer to the sink, the difference between the strategies widens. The justification of the differences relies in the number of packets generated and forwarded by each strategy (more packets demand more energy for radio transmission) and also on the fact that by having more packets being transmitted, the interferences are also higher, leading to frequent aborted transmissions. Fig. 5-15 analysis the number of packets generated, forwarded and lost.

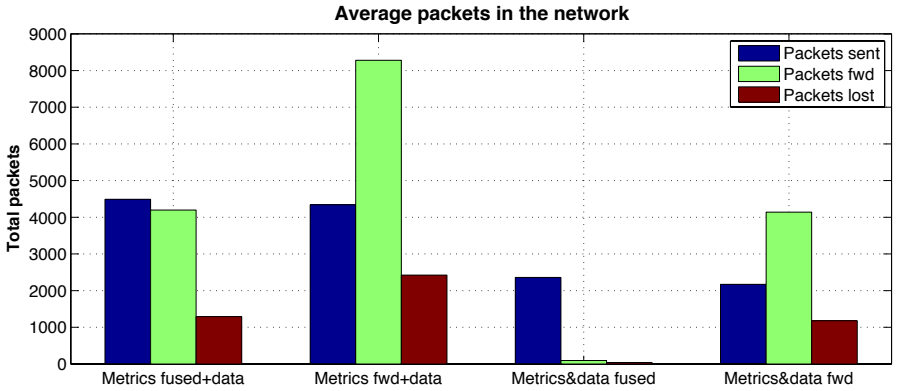


Fig. 5-15 – Scenario 1: Average number of packets in the network.

Using the formulas described in Appendix A (Table A-3), the number of expected packets (sent and forwarded) for Scenario 1 in each 6 s period is of 93 for strategies that

use no fusion and 31 for those that use fusion. In the strategies that also generate separated data packets, another 93 have to be added. This totalizes 8680 packets for Strategy 1 $((31\ performance\ packets + 93\ data\ packets)\times 70\ reporting\ cycles)$, 13020 for Strategy 2 $((93+93)\times 70)$, 2170 for Strategy 3 (31×70) and 6510 for Strategy 4 (93×70). The results show an average number of messages lower than the prediction. This is due to the fact that some messages are lost and that in the beginning of the tests there were no messages to forward. As can be seen, the number of forwarded packets is the main reason for the overhead presented by Strategy 2 (*Metrics fwd + data*). The strategy *Metrics fused + data* is the one that generates more packets (by a small amount) since it also sends alert packets. The number of packets by node is analysed in Fig. 5-16.

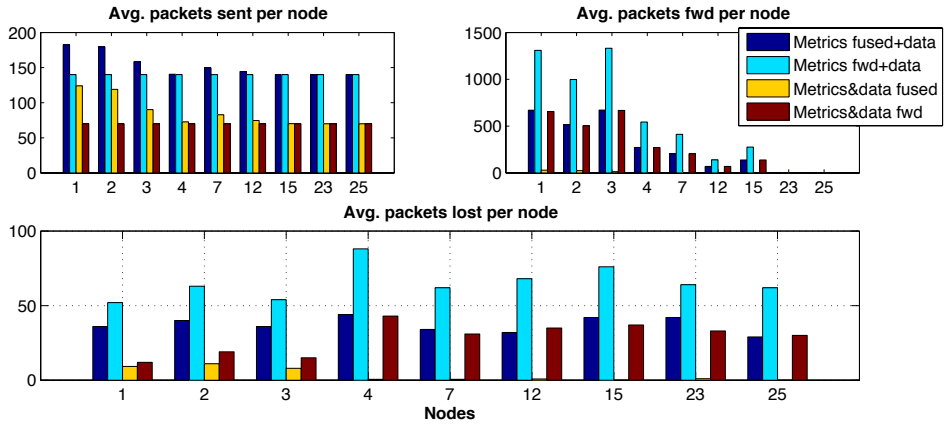


Fig. 5-16 – Scenario 1: Average number of packets sent, forwarded and lost by strategy used.

With a direct correlation to the values of energy spent presented in Fig. 5-14, the number of messages that each node has to process increases, as the node is closer to the sink. Also, the interferences caused by the high number of messages produced causes the loss of many messages. In Fig. 5-16 lost packets presented are the sum of packets sent originally from the referenced node that did not arrive destination. In fact, most of the drops that are presented in the leaf nodes of the topology are in fact losses that occurred while other nodes were forwarding the messages and not by the actual transmission made by original sending nodes.

The average reliability of the network was also calculated based on the packet delivery ratio, given the number of packets received and sent by each node. Its results, together with the maximum, minimum and standard deviation of the set of tests used, are depicted in Fig. 5-17.

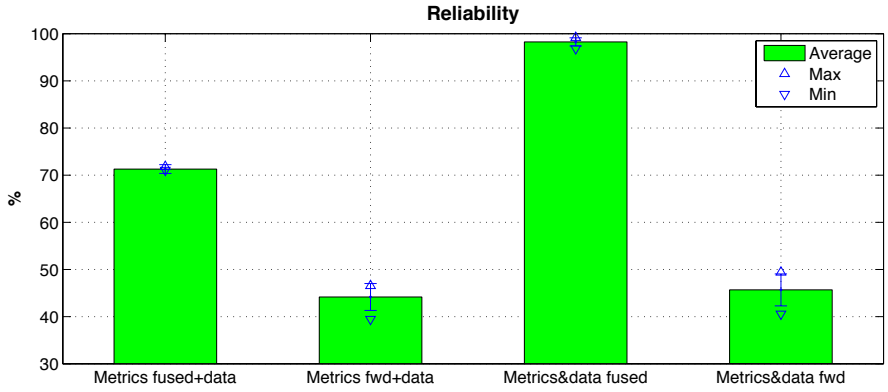


Fig. 5-17 – Scenario 1: Network reliability by strategy used.

Comparing the reliability results from the four different strategies it is clear that there is a big discrepancy in the numbers presented, due to the effects of the packet loss (Strategy 3 achieves an average of 98.26% and Strategy 2 achieves 44.15%). This difference remains stable across all tests (standard deviation, as well as the maximum and minimum values obtained, are all in a close range). However, the comparison is unfair. In fact, when using WMCAP most of the transmissions are made between 2 neighbour nodes (only alerts go directly to the sink), what makes it easier to prevent losses, especially in network areas with less traffic congestion (and that are not dependent on forwarding in more congested areas). When using strategies that do not include WMCAP, all messages are sent end-to-end (from the original sender to the sink), being subject to problems along the all path to the sink. To better assess the actual results, and effectively compare the reliability of each approach, the terms of comparison must be changed. So, instead of comparing the raw values of packets sent, received and lost, the comparison will be made considering the information that actually arrives to the sink, the one that effectively counts for the monitoring of the network by the network manager. In this case, all the reception of packets by intermediate nodes will not be considered. As each metric report packet contains the number of samples that constitutes each metric reported (see Appendix A), this value will be used to compare the effective report of metrics that reach the sink. In fact, each sample can be counted as information from a different node (that corresponds to a *METRIC_REPORT_FUSION* packet that was sent from the same node), which was fused along the path to the sink. In this study only the samples of one of the metrics transported are counted (every packet may carry several metrics). The goal is to count the number of nodes involved in the generation of each metric report packet. Alerts and packets of regular data are already counted as end-to-end. The results based on this approach gave the following results (Fig. 5-18 and Fig. 5-19).

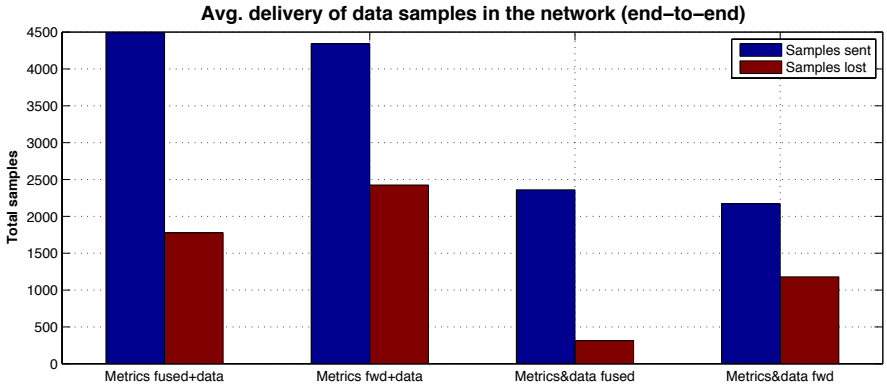


Fig. 5-18 – Scenario 1: Average number of samples delivered to the sink.

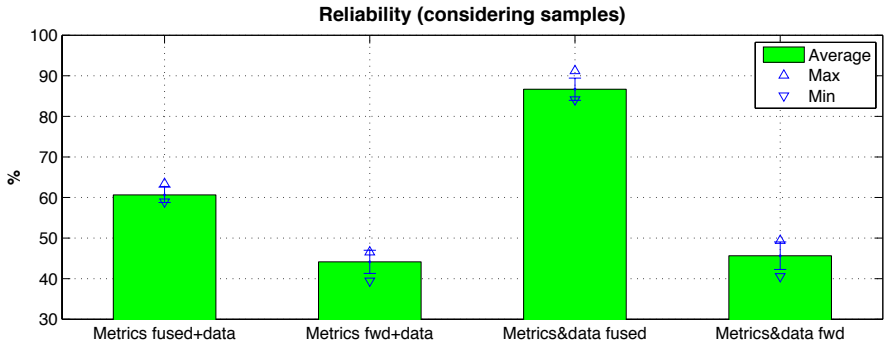


Fig. 5-19 – Scenario 1: Network reliability considering data samples delivered to the sink.

From the results it can be seen that while continuing to achieve better results when compared to other strategies, both strategies that use fusion have a real reliability that is lower than the one calculated before (strategies that use no fusion do not have their values changed). Strategy 1 lowered from 71.28% to 60.65% while Strategy 3 lowered from 98.26% to 86.70%). The higher impact of loosing a fused message is also clear. Although the scenarios where Data & Performance were fused presented a lower packet loss than the forwarded ones, the loss of one of those packets has a big impact if samples of individual information are considered.

Scenario 1 presented a network with congestion what resulted in high packets losses in most of the scenarios. Scenario 2, on the other hand, presents a planned network where data rates and synchronization between nodes was thought to avoid most of the congestion due to interferences under normal conditions. Next figures show how Strategies 1 and 2 compare in terms of energy spent and number of packets.

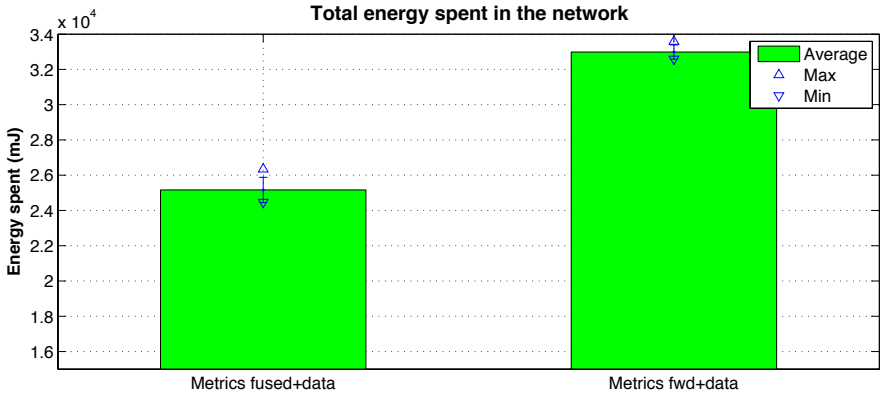


Fig. 5-20 – Scenario 2: Total energy spent in the network by strategy used (average).

The fact that Fig. 5-20 presents a lower waste of energy in both strategies, when compared to Scenario 1, in not surprising if considering the reduced number of packets that are transmitted - only leaf nodes send regular data packets in this scenario (Fig. 5-21, Fig. 5-22 and Fig. 5-23).

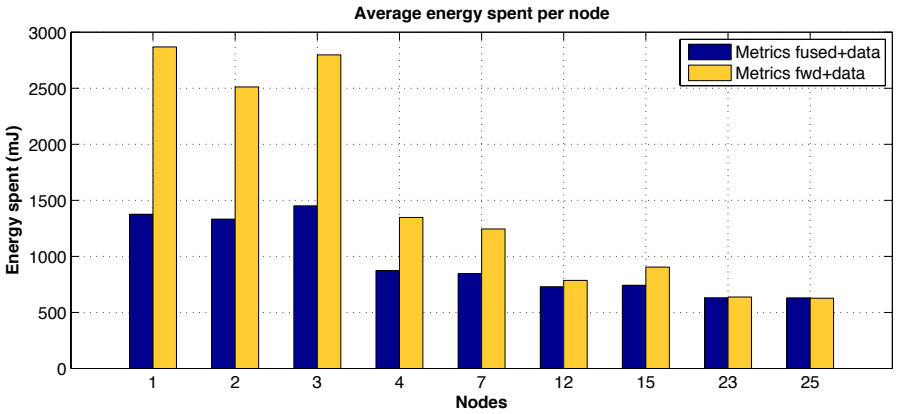


Fig. 5-21 – Scenario 2: Energy spent per node by strategy used.

Using the formulas described in Appendix A (Table A-3), the number of expected performance packets (sent and forwarded) for Scenario 2 in each 6 s period is 31 for Strategy 1 (Metrics fused + data) and 93 for Strategy 2 (Metrics fwd + data, that uses no fusion). The number of expected regular data packets in each 9 s period is 48 (in Scenario 2 only leaf nodes send regular data packets each 9 s, which are then forwarded by upper nodes during 46 reporting cycles). This totalizes 4378 packets for Strategy 1 ($31 \times 70 + 48 \times 46$) and 8718 for Strategy 2 ($93 \times 70 + 48 \times 46$) – data packets only have 46 reporting cycles as the simulation finishes when performance packets reach 70 reporting cycles. The results show an average number of messages lower than the prediction for the same reasons of Scenario 1.

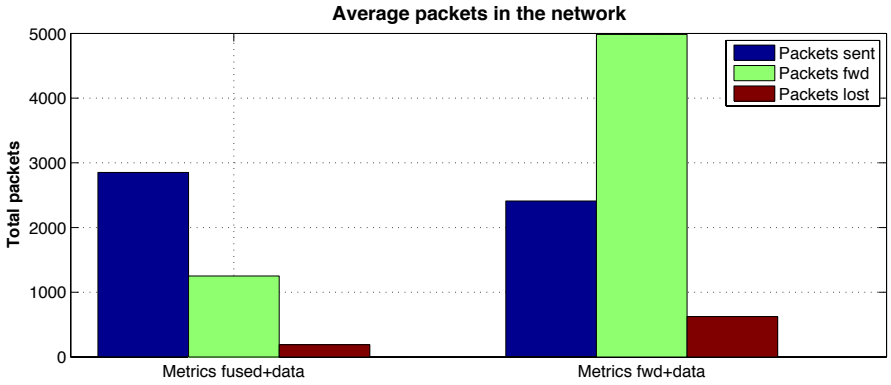


Fig. 5-22 – Scenario 2: Average number of packets in the network during all simulation time.

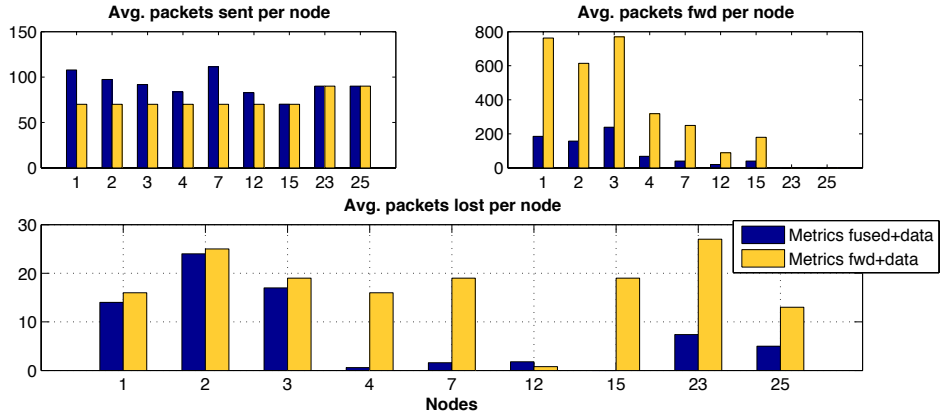


Fig. 5-23 – Scenario 2: Average number of packets sent, forwarded and lost by strategy used during.

Due to the lower number of packets loss, reliability increased for both scenarios (Fig. 5-24).

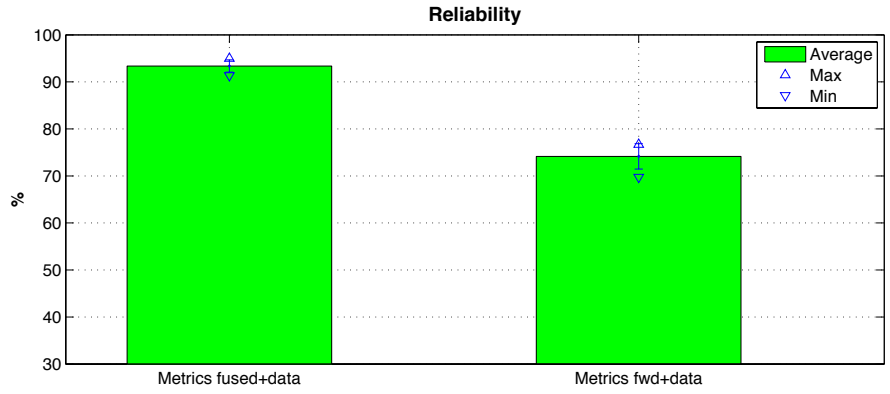


Fig. 5-24 – Scenario 2: Network reliability by strategy used.

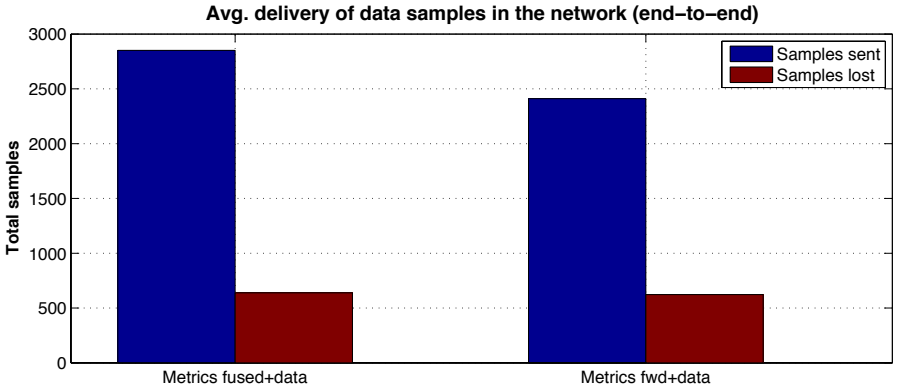


Fig. 5-25 – Scenario2: Average number of samples delivered to the sink.

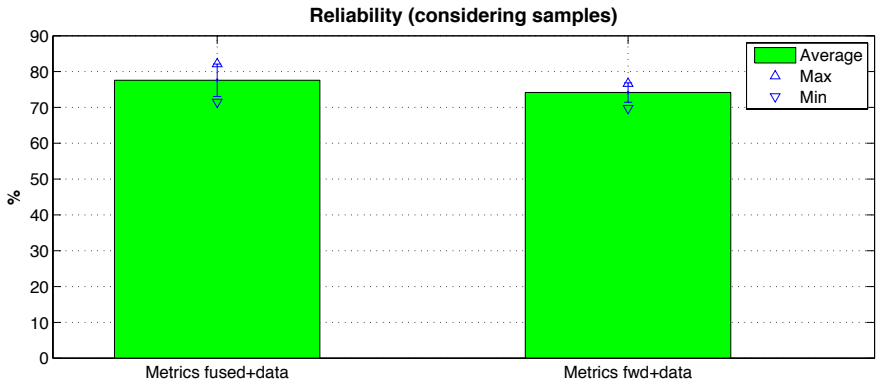


Fig. 5-26 – Scenario 2: Network reliability considering data samples delivered to the sink.

When considering samples in reliability (Fig. 5-25 and Fig. 5-26) a slight advantage is still obtained by the metrics fusion strategy (77.58% versus 74.16%). The reason to the lower difference relates to the fact that, while both the spent energy and packet loss are lower when using fusion (Fig. 5-20 and Fig. 5-22), the impact of a lost metric report packet is greater than the loss of a single metric message.

Another impact to consider when using the proposed fusion protocol is the freshness of the sample values. While in the approaches that forward metric packets, the metric values are as fresh as the time they take to arrive the sink, being only dependent on the number of hops and on the transmission conditions along the path, when considering fusion a relay is imposed between each hop. The relay is necessary to allow for the arrival of metrics information from downstream neighbours, but imposes a fixed delay between each hop. Next figures (Fig. 5-27 and Fig. 5-28) analyse the samples delay (end-to-end) using Scenario 3 (scenario without regular data).

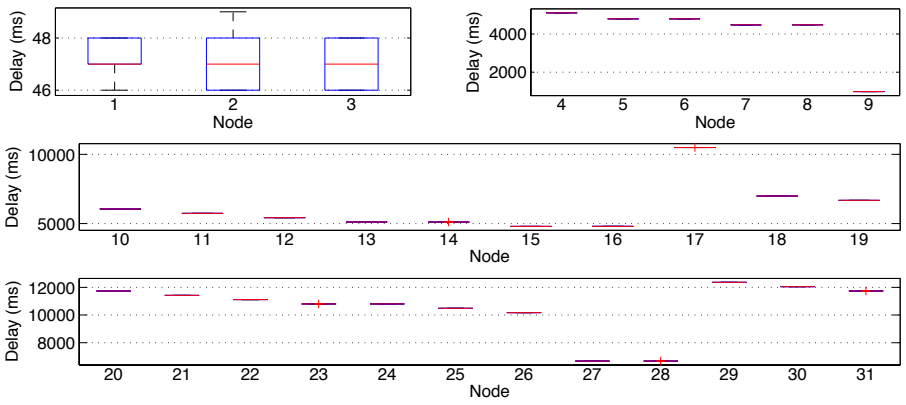


Fig. 5-27 – Scenario 3: Metrics fused and no regular data - Delay of samples from original sender to sink.

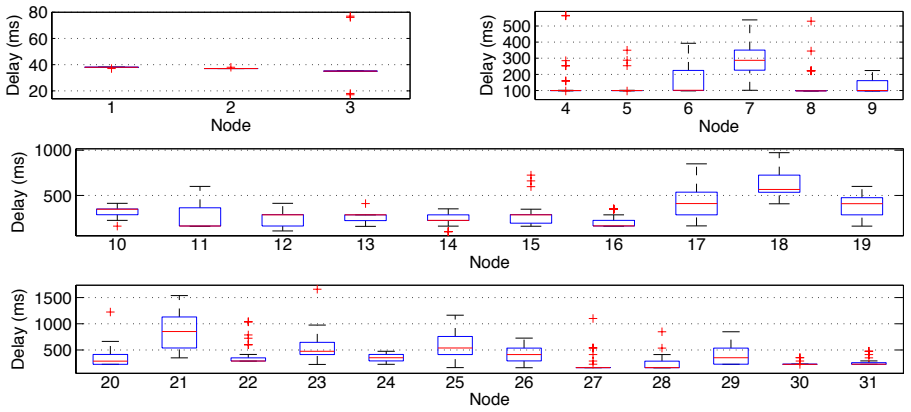


Fig. 5-28 - Scenario 3: Metrics forwarded and no regular data - Delay of samples from original sender to sink.

The impact of the relay imposed to fused data is clear in the results shown, where nodes are grouped by their level in the network tree. The difference in delays between nodes of the same level, when using WMCAP, can be explained by the times at which nodes send data. As each node started the sequence of transmissions of performance data at different times (and only after that time with fixed intervals of 6 seconds) their data may spend more or less time in following upstream nodes (Fig. 5-29). In Scenario 3, if considering no interference, a node that transmits its performance values at time t_x has its samples retransmitted by the next upstream neighbour immediately as the samples arrive (2nd line of the figure) or at most at time t_x+6s (3rd line of the figure).

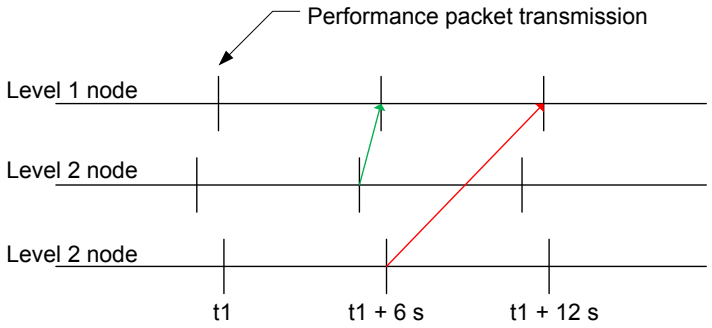


Fig. 5-29 – Transmission of a performance packet to the next upstream node.

The relay imposed by the protocol does not affect any alerts, only metrics within healthy bounds. The relay is also dependent both on the cadence that is set in the protocol and on the radio duty cycle of the node.

5.7 EVALUATION ANALYSIS OVERVIEW

As the results show, the proposed protocol is both energy and packet-efficient. When compared to a solution that implies the transmission of individual performance concurrently with regular data, or performance data within regular data packets, it presents lower energy consumption and uses fewer packets. By saving energy it provides for a lower overhead and contributes to a longer lifetime of the nodes. By reducing the number of packets needed to transport the metrics it contributes to lower interferences and fewer retransmissions due to collisions, which not only save energy but also contribute to reduce the performance degradation that measuring the performance necessarily brings to the network. The protocol can be used in two ways. The first implies the use of performance packets fused along the network and is suited to scenarios where not all nodes send regular data, or where all nodes send regular data but at a different cadence of performance update needs, or when regular and performance packets report to different sinks. The second extends the functionality of metrics transmission to regular data, by allowing that both performance data and regular data are sent within the protocol packets, being suited to scenarios where performance and regular data share the same cadence of reporting and similar fusion needs. By using one of these options, the overhead implied by a network with performance control is limited, providing for a constant monitoring of the global performance of the network. The use of alerts allow for control of specific individual metric values. As a drawback of the protocol, is the fact that in spite of having an adjustable cadence, the performance collection using the protocol does not give a picture of the global network in a specific time, as the samples that arrive the sink are from different moments in the network, the moments when they were collected in their source nodes. However, this is not a real necessity in real deployments, especially if considering the existence of alarms for every threshold broke by an individual metric. In our view, what is of crucial importance to the network manager is to be able to have a constant knowledge of the global performance of the network, being

assured that if a malfunction occurs an alert is immediately generated and sent to the sink. By using the proposed protocol, an update of the global QoSensing of the network is received at each metric reporting interval.

In conclusion, the protocol proposed is designed to work in WSNs with controlled performance, especially those that are carefully planned, with reduced overhead. It can use any type of metrics that can be fused using decomposable functions, being a generic solution that can be used for any kind of QoSensing monitoring. Also, it is a lightweight solution, not only because of the reduced overhead in the network but also because it implies a reduced maintenance, and scarce resources.

5.8 CHAPTER SUMMARY

In this chapter, the collection of performance data was focused. Data fusion methods were analysed as a method to reduce both energy and the number of packets in the network when gathering all the performance packets produced in the nodes. A new general classification of data gathering protocols in a WSN perspective, which includes the already existent, was proposed. A new protocol to deal with the specificities of WSNs monitoring, using data fusion, was also proposed. The protocol was specified (further details are explained in Appendix A) and evaluated using simulation. When compared to solutions that use no fusion it presents better results both in the energy used as in the number of packets generated in the network.

Chapter

6

Conclusions and Future work

This chapter presents an overview of the work done during the thesis, highlights its contribution and also gives possible directions for the continuation of the work produced.

6.1 OVERVIEW

The expansion of WSNs to performance demanding scenarios and application opens both new opportunities and new challenges. The opportunities rely on the use of the flexibility and reduced cost of these networks to enable a vast set of new applications. The challenges are in measuring, controlling and assuring the needed performance.

WSNs with controlled performance are the main target of the work developed in this thesis. While all the work may be applied to any type of WSN, it is in networks that need to run critical applications, or at least where performance is an issue, that it best applies. The inexistence of a common framework to address the performance in WSNs led to multiple proprietary approaches, and to specifically tailored applications and protocols. While this situation can be maintained if only a selected number of scenarios and applications use WSNs with controlled performance, it prevents its general use and understanding. Also, the development of metrics to evaluate the performance mostly follows the rule of necessity, with metrics being created without a supporting common framework.

By having a common taxonomy of requirements and a framework of metrics that cover the needs of WSNs with controlled performance, it is possible to provide a common language that permits a more accurate definition of the requirements of the WSN. As a result, it facilitates the specification and planning of the network by giving values to the performance required (before deployment), enables the evaluation and test of the network during deployment, by comparing its performance with the values predefined for the metrics, and enables the continuous monitoring of the network by using the metrics and healthy ranges predefined (after deployment, during operation phase).

This thesis starts by analysing the special characteristics of WSNs and of the specifics of its performance needs. For this analysis, the participation in project GINSENG was used as a case study on the requirements of an industrial WSN with their multiple scenarios. The lack of a common framework that allowed for the specification of the required performance in a WSN, led to the study of taxonomies and to the proposal of new classifications. The primary objective of this approach was to understand which were the performance requirements of a WSN. The second was to create a support on which metrics that evaluate those requirements could be created in a coherent manner. The specific characteristics that metrics for WSNs should have, due to the multiple restrictions of these networks, were then addressed. As a result, a global framework of metrics and a specific one that addressed performance in industrial scenarios were proposed. The use of metrics to dynamically control the maintenance of a predetermined level of performance was also addressed. To effectively address performance, it is required that common nodes send performance information to the base station, where a deeper analysis can be made and where corrective actions can be triggered (corrective actions can also be made locally in nodes but are more limited). The level of performance traffic generated, especially

when a real-time evaluation is necessary, may be difficult to sustain, and may degrade the same performance that it is measuring. To overcome that problem, the use of fusion applied to the necessities of performance data was considered and studied. As a result, a new generic protocol was proposed.

6.2 CONTRIBUTIONS

The main contributions of this thesis work were already presented in the Introduction. In this section, all contributions are reviewed and discussed.

A new classification of WSNs applications scenarios, which combines and upgrades prior classifications to include time-critical applications, was presented. This classification addresses WSNs from the point of view of application scenarios and was made as a starting point to the analysis of the requirements of a WSN. This classification was applied to known applications for validation.

A new taxonomy for the QoSensing requirements of a WSN was proposed. This taxonomy tries to accurately describe WSNs in the point of view of the user or application, focusing in their QoSensing requirements. By using the taxonomy as a common reference model, the needs of any WSNs in terms of QoSensing can be specified and understood by all the participants, from network managers to final users. This will apply before the deployment of a new network, by allowing an easy specification of the requirements that need to be addressed, during deployment, by enabling easier validation tests, and also in operation time by providing a clear set of requirements to monitor (by evaluating their respective metrics). Also, it creates a reference model from which new metrics can be created. These metrics will fulfil each of the found requirements in the taxonomy and provide for their effective evaluation. The taxonomy was applied to the requirements of known applications for validation. An initial version of the taxonomy was also presented to the ISO/IEC JTC 1 study group on Sensor Networks and included in the Technical report produced.

New set of metrics and a specific framework. In order to understand which type of metrics were more adequate to WSNs, especially considering real-time necessities, a study on different types of metrics, individual, collective and composed (aggregated metrics were used), was accomplished. These metrics were evaluated by simulation to find their advantages and disadvantages when applied to the assessment of the QoSesing of a WSN. Collective metrics were found to be a good option in the evaluation of the global QoSensing of a WSN with controlled performance, by hiding the normal fluctuations of values associated to this networks (to both metrics and data), requiring a reduced number of performance data transmissions and by giving a context to individual metrics. A set of rules to consider when creating metrics to WSNs was also presented. The aim is that the creation of metrics always considers the specific characteristics of WSNs. A new set of metrics that evaluates the requirements of the Network performance

branch of the taxonomic tree of requirements (presented before) was also proposed. The choice to address this branch was taken because it contains the requirements that were found essential in a WSN with controlled performance in industrial scenarios. The metrics proposed were divided by the network phase where they are more useful (deployment, operation, debug&recovery) and uses individual, aggregated and collective metrics. A specific and reduced framework dedicated to WSNs in industrial scenarios was also proposed, based on the priority requirements of the three main subsystems that are used in an Industrial-Manufacturing sector network (the indicator system, the semi-automatic control system and the automatic control system). The control and maintenance of the levels of performance during operation, by controlling QoSensing metrics, was also addressed with the tests done for DynMAC. While the protocol is just an example of how metrics can be used to enable a dynamic control of the performance along the life cycle of a WSN, it proved useful to upgrade the resilience of an already performance-targeted MAC protocol. A new general classification of data gathering protocols, based on network organization and in a WSN perspective, was also proposed, as a small contribution that builds on, and summarizes, existing previous classifications.

A new protocol for metrics collection that uses data fusion and considers the specific needs of performance monitoring in WSNs was also proposed. The protocol uses collective metrics to assess the global QoSensing of a WSN, together with specific alerts that can deal with any specific performance situation. The protocol was tested by simulation and presented good results in terms of energy spent and number of packets generated when compared to typical solutions, while providing for an effective evaluation of the QoSensing of a WSN.

The thesis also contributed to the **GINSENG** project, through the participation of the author in Task 1.2- “Measure of performance”. The contributions made were in the classification of the project scenarios, according to the classifications (classification of applications and taxonomy of WSN requirements) developed by the author, and in the specification of the metrics to use.

6.3 FUTURE WORK

The work produced for this thesis tried to respond to the needs of a WSN with performance control. However, the contributions made can be further developed and can also open the door to further contributions in the same area, specifically in the main three areas addressed: taxonomy, QoSensing metrics and metrics collection.

The taxonomy for the QoSensing requirements of a WSN, made from a user/application perspective, can be further potentiated if used to classify specific network components such as protocols, mechanisms or technologies, each responsible for providing a specific part of the global QoSensing. By comparing user/application requirements with a set of

previously classified network components, an automatic matching can be possible, being a powerful tool to help network managers in their deployment decisions.

To achieve a complete evaluation of the global QoSensing of a network, in all its dimensions, the development of metrics must be completed for all the branches of the taxonomic tree of requirements that were not addressed. Each of the metrics must be able to evaluate each taxon and, in order to comply to the specific WSN restrictions and needs, should be specified following the approach proposed in this thesis.

Further tests can be made to the metrics collection protocol proposed. A study of the protocol performance correlating number of hops from the sink, protocol update time and alerts warning thresholds, to assess overhead and accuracy of the results obtained (when compared to a typical end-to-end performance delivery) would be valuable as an indication of the protocol suitability to different types of deployments and application. Also, in this thesis only the base version of the metrics collection protocol was tested. Further tests are needed to evaluate the extended version proposed, which allows for additional flexibility in the metrics collection process while implying more overhead. An analysis of the benefits and advisable scenarios for the use of each of the protocol versions must be further analysed. A study on the protocol performance under dynamic topologies, while not the typical scenario for planned WSNs with controlled performance, could also assess the application of the protocol to additional scenarios.

References

[1] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. Chichester, UK: John Wiley & Sons, Ltd, 2005.

[2] V. Gupta and R. Pandey, "Data Fusion and Topology Control in Wireless Sensor Networks," *WSEAS Trans. Signal Process.*, vol. 4, no. 4, pp. 150–172, 2008.

[3] A. Boukerche, R. B. Araujo, and L. Villas, "A Wireless Actor and Sensor Networks QoS-Aware Routing Protocol for the Emergency Preparedness Class of Applications," in *31st IEEE Conference on Local Computer Networks*, 2006.

[4] D. Chen and P. K. Varshney, "QoS Support in Wireless Sensor Networks : A Survey," in *Proc. of the 2004 International Conference on Wireless Networks (ICWN 2004)*, 2004.

[5] "ICT FP7 0384239 Ginseng - Performance Control in Wireless Sensor Networks." [Online]. Available: ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/necs/fp7-fact-sheet-ginseng_en.pdf. [Accessed: 01-Jun-2015].

[6] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "RFC 2330: Framework for IP Performance Metrics." 1998.

[7] M. Alves, "The WSN standards and COTS landscape: can we get QoS and 'calm technology'?," *6th European Conference on Wireless Sensor Networks (EWSN 2009) - Tutorial*. 2009.

[8] M. Weiser and J. S. Brown, "Designing Calm Technology." Xerox PARC, 1995.

[9] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, 2002.

[10] V. Raïsänen, *Implementing Service Quality in IP Networks*. John Wiley & Sons, England, 2003.

[11] "Wikipedia - Quality of Service." [Online]. Available: http://en.wikipedia.org/wiki/Quality_of_service. [Accessed: 05-Jan-2015].

[12] F. Xia, "QoS Challenges and Opportunities in Wireless Sensor/Actuator Networks," *Sensors 2008*, vol. 8, pp. 1099–1110, 2008.

[13] J. Martínez, A.-B. García, I. Corredor, L. López, V. Hernández, and A. Dasilva, "QoS in Wireless Sensor Networks : survey and approach," in *Euro American conference on Telematics and Information Systems*, 2007.

[14] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, 2003.

[15] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Pers. Commun.*, vol. 7, no. 5, pp. 16 – 27, 2000.

[16] E. C. H. Ngai and M. R. Lyu, "A Real-Time Communication Framework for Wireless Sensor-Actuator Networks," in *IEEE Aerospace Conference*, 2006.

[17] "ANSI/ISA-100.11a-2011 Wireless systems for industrial automation: Process control and related applications." [Online]. Available: www.isa.org/isa100. [Accessed: 01-Jan-2014].

[18] "HART Communication Foundation, WirelessHART." [Online]. Available: http://hartcomm.org. [Accessed: 01-Jan-2014].

[19] "International Society of Automation (ISA)." [Online]. Available: http://www.isa.org. [Accessed: 05-Jan-2015].

[20] "GINSENG - Performance Control in Wireless Sensor Networks - Description of Work (DoW) - FP7-ICT-2007-2, Project 224282," 2008.

[21] "D1.3 - Final GINSENG Architecture, Scenarios and Quality of Service Measures." FP7 GINSENG (INFSO-ICT-224282) Deliverable D1.3, WP1, 2010.

[22] N. Kushalnagar, G. Montenegro, and C. Schumacher, "RFC 4919: IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem

Statement, and Goals." IETF Network Working Group, Informational, pp. 1–12, 2007.

[23] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - A lightweight and flexible operating system for tiny networked sensors," in *First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, 2004.

[24] A. Dunkels, "Rime-a lightweight layered communication stack for sensor networks," in *European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, 2007.

[25] "D1.1 - GINSENG Architecture, Scenarios and Quality of Service Measures." FP7 GINSENG (INFSO-ICT-224282) Deliverable D1.1, WP1, 2009.

[26] "Taxonomy," *Encyclopedia Britannica*. [Online]. Available: http://www.britannica.com/EBchecked/topic/584695/taxonomy. [Accessed: 06-Jun-2015].

[27] "Technical Document of ISO/IEC JTC 1 - Study on Sensor Networks (Version 3)." Study Group on Sensor Networks, 2009.

[28] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," *ACM Mob. Comput. Commun. Rev.*, vol. 6, no. 2, pp. 28–36, 2002.

[29] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy, "The platforms enabling wireless sensor networks," *Commun. ACM*, vol. 47, no. 6, pp. 41–46, 2004.

[30] S. Cheekiralla and D. W. Engels, "A functional taxonomy of wireless sensor network devices," in *2nd International Conference on Broadband Networks, BROADNETS 2005*, 2005, vol. 2005, pp. 26–33.

[31] K. Römer and F. Mattern, "The design space of wireless sensor networks," *IEEE Wirel. Commun.*, vol. 11, no. 6, 2004.

[32] V. Rocha and Gil Gonçalves, "Sensing the World : Challenges on WSNs," in *Proc. of the IEEE Inernacional Conference on Automation, Quality and Testing, Robotics*, 2008, pp. 1–13.

[33] A. Iyer, S. S. Kulkarni, V. Mhatre, and C. P. Rosenberg, "A Taxonomy-based Approach to Design of Large-scale Sensor Networks," in *Wireless Sensor Networks and Applications*, Y. Li, M. T. Thai, and W. Wu, Eds. Springer, 2008.

[34] R. Mac Ruairí, M. T. Keane, and G. Coleman, "A Wireless Sensor Network Application Requirements Taxonomy," in *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*, 2008, pp. 209–216.

[35] L. . S. Bai, R. . P. Dick, and P. A. Dinda, "Archetype-based design: Sensor network programming for application experts, not just programming experts," in *Proc. of 2009 International Conference on Information Processing in Sensor Networks (IPSN 2009)*, 2009, pp. 85–96.

[36] V. Pereira, J. S. Silva, J. Granjal, R. Silva, E. Monteiro, and Q. Pan, "A taxonomy of Wireless Sensor Networks with Qos," in *Proc. of 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2011.

[37] L. Mottola and G. Pietro Picco, "Programming Wireless Sensor Networks: Fundamental Concepts and State of the Art," *ACM Comput. Surv.*, vol. 43, no. 3, pp. 1–51, 2011.

[38] F. J. Oppermann, C. A. Boano, and K. Romer, "A Decade of Wireless Sensing Applications: Survey and Taxonomy," in *The Art of Wireless Sensor Networks - Volume 1: Fundamentals*, H. M. Ammari, Ed. Springer, 2014.

[39] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, "A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking," *Comput. Networks Int. J. Comput. Telecommun. Netw. - Spec. issue Mil. Commun. Syst. Technol.*, vol. 46, no. 5, pp. 1–27, 2004.

[40] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking:design tradeoffs and early experiences with ZebraNet," in *Conference on Architectural Support for Programming Languages and Operating Systems*, 2002.

[41] R. Silva, J. Sá Silva, and F. Boavida, "A proxy-based mobility solution for critical WSN applications," in *2010 IEEE International Conference on Communications Workshops, ICC 2010*, 2010.

[42] W. Wang, V. Srinivasan, and K.-C. Chua, "Trade-offs between mobility and density for coverage in Wireless Sensor Networks," in *Proc. of the 13th annual ACM international conference on Mobile computing and networking - MobiCom '07*, 2007.

[43] E. Ekici, Y. Gu, and D. Bozdag, "Mobility-Based Communication in Wireless Sensor Networks," *IEEE Commun. Mag.*, vol. 7, no. July, pp. 56–62, 2006.

[44] "Applications of Wireless Sensor Networks in Next Generation Networks." ITU-T Technical Paper, Series Y.2000: Next Generation Networks, 2014.

[45] Y. Cherdantseva and J. Hilton, "A Reference Model of Information Assurance & Security," in *2013 International Conference on Availability, Reliability and Security*, 2013, pp. 546–555.

[46] M. E. Whitman and H. J. Mattord, *Principles of Information Security*, 4th ed. Cengage Learning, 2011.

[47] S. Convery, "Network Authentication, Authorization, and Accounting Part One: Concepts, Elements, and Approaches," *Internet Protoc. Journal, Cisco Syst. Inc*, vol. 10, no. 1, pp. 2–11, 2007.

[48] Z. Taghikhaki, N. Meratin, and P. J. M. Havinga, "On QoS Guarantees of Error Control Schemes for Data Dissemination in a Chain-based Wireless Sensor Networks," *Sensors Transducers, Spec. issue*, vol. 18, pp. 188–202, 2013.

[49] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis, "Design and deployment of industrial sensor networks: Experiences from a semiconductor plant and the north sea," in *Proceedings of the SenSys'05*, 2005, pp. 64–75.

[50] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli, "Wireless Sensor Networks for Environmental Monitoring: : The SensorScope Experience," in *2008 IEEE International Zurich Seminar on Communications*, 2008.

[51] "International Standard, Information processing systems – Open Systems Interconnection – Basic Reference Model, Part 4: Management framework." Reference Number: ISO/IEC 7498 - 4 : 1989 (E), 1989.

[52] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, and Et.Al., "VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance," *ACM Trans. Sens. Networks*, vol. 2, no. 1, 2006.

[53] S. M. Abd El-Kader and B. M. Mohammad El-Basioni, "Precision farming solution in Egypt using the wireless sensor network technology," *Egypt. Informatics J.*, vol. 14, no. 3, pp. 221–233, 2013.

[54] A. Morton and S. Van den Berghe, "RFC 5835: Framework for Metric Composition." Internet Engineering Task Force (IETF), Informational, pp. 1–19, 2010.

[55] G. Martinovic, J. Balen, and D. Zagar, "A Cross-Layer Approach and Performance Benchmarking in Wireless Sensor Networks," in *Proceedings of the 2nd WSEAS International Conference on Sensors, and Signals and Visualization, Imaging and Simulation and Materials Science*, 2009, pp. 76–81.

[56] A. Sen, B. H. Shen, L. Zhou, and B. Hao, "Fault-Tolerance in Sensor Networks : A New Evaluation Metric," in *Proc. of INFOCOM 2006*, 2006.

[57] N. Ahmed, Y. Dong, S. S. Kanhere, S. Jha, M. Rutten, T. Bessell, and N. Gordon, "Performance evaluation of a Wireless Sensor Network based tracking system," in *2008 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, 2008, pp. 163–172.

[58] I. Dietrich and F. Dressler, "On the lifetime of wireless sensor networks," *ACM Trans. Sens. Networks*, vol. 5, no. 1, pp. 1–38, 2009.

[59] T. O'Donovan, N. Tsiftes, Z. He, T. Voigt, and C. J. Sreenan, "Detailed diagnosis of performance anomalies in sensornets," in *Proceedings of the 6th Workshop on Hot Topics in Embedded Networked Sensors - HotEmNets '10*, 2010, p. 1.

[60] B. Krishnamachari, D. Estrin, and S. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," in *Proc. of the 22nd International Conference on Distributed Computing Systems Workshops*, 2002.

[61] D. Wagner, "Resilient aggregation in sensor networks," in *Proc. of the 2nd ACM workshop on Security of ad hoc and sensor networks - SASN '04*, 2004, p. 78.

[62] V. Pereira, J. S. Silva, and E. Monteiro, "A framework for Wireless Sensor Networks performance monitoring," in *Proc. of 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2012.

[63] Y. Lingyun and W. Xingchao, "Study on performance evaluation method based on measurement for Wireless Sensor Network," in *International Conference on Communications Technology and Applications (ICCTA)*, 2009, pp. 201–206.

[64] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock Synchronization for Wireless Sensor Networks : A Survey," *Ad Hoc Networks*, vol. 3, 2005.

[65] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE Netw.*, vol. 18, no. 4, pp. 45–50, 2004.

[66] M. L. Sichitiu and C. Veerarittiphan, "Simple, Accurate Time Synchronization for Wireless Sensor Networks," *Proc. WCNC*, 2003.

[67] P. Sommer and R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks," *2009 Int. Conf. Inf. Process. Sens. Networks*, 2009.

[68] J. Fabini and A. Morton, "RFC 7312: Advanced Stream and Sampling Framework for IP Performance Metrics (IPPM)." Internet Engineering Task Force (IETF), Network Working Group, Informational, 2014.

[69] J. Mahdavi and V. Paxson, "RFC 2678: IPPM Metrics for Measuring Connectivity." Internet Engineering Task Force (IETF), Network Working Group, Standards Track, 1999.

[70] G. Almes, S. Kalidindi, and M. Zekauskas, "RFC 2679: A One-way Delay Metric for IPPM." 1999.

[71] G. Almes, S. Kalidindi, and M. Zekauskas, "RFC 2680: A One-way Packet Loss Metric for IPPM." 1999.

[72] G. Almes, S. Kalidindi, and M. Zekauskas, "RFC 2681: A Round-trip Delay Metric for IPPM." 1999.

[73] A. Morton, "RFC 6673: Round-Trip Packet Loss Metrics." Internet Engineering Task Force (IETF), Network Working Group, Standards Track, 2012.

[74] R. Koodli and R. Ravikanth, "RFC 3357: One-way Loss Pattern Sample Metrics." 2002.

[75] C. Demichelis and P. Chimento, "RFC 3393: IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)." Internet Engineering Task Force (IETF), Network Working Group, Standards Track, 2002.

[76] V. Raisanen, G. Grotefeld, and A. Morton, "RFC 3432: Network performance measurement with periodic streams." Internet Engineering Task Force (IETF), Network Working Group, Standards Track, 2002.

[77] A. Morton, L. Ciavattone, G. Ramachandran, S. Shalunov, and J. Perser, "RFC 4737: Packet Reordering Metrics." Internet Engineering Task Force (IETF), Network Working Group, Standards Track, 2006.

[78] M. Mathis and M. Allman, "RFC 3148: A Framework for Defining Empirical Bulk Transfer Capacity Metrics." Internet Engineering Task Force (IETF), Network Working Group, Informational, 2001.

[79] P. Chimento and J. Ishac, "RFC 5136: Defining Network Capacity." 2008.

[80] H. Uijterwaal, "RFC 5560: A One-Way Packet Duplication Metric." Internet Engineering Task Force (IETF), Network Working Group, Standards Track, 2009.

[81] E. Stephan, L. Liang, and A. Morton, "RFC 5644: IP Performance Metrics (IPPM): Spatial and Multicast." Internet Engineering Task Force (IETF), Network Working Group, Standards Track, pp. 1–57, 2009.

[82] V. Paxson, "Towards a Framework for Defining Internet Performance Metrics," in *Proceedings of INET '96*, 1996.

[83] "RFC 3763: One-way Active Measurement Protocol (OWAMP) Requirements." Internet Engineering Task Force (IETF), Network Working Group, Informational, 2004.

[84] S. Shalunov, B. Teitelbaum, A. Karp, J. Boote, and M. Zekauskas, "RFC 4656: A One-way Active Measurement Protocol (OWAMP)." Internet Engineering Task Force (IETF), Network Working Group, Standards Track, 2006.

[85] K. Hedayat, R. Krzanowski, A. Morton, K. Yum, and J. Babiarz, "RFC 5357: A Two-Way Active Measurement Protocol (TWAMP)." Internet Engineering Task Force (IETF), Network Working Group, Standards Track, 2008.

[86] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wirel. Commun.*, vol. 14, no. 2, 2007.

[87] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *Proceedings of the 4th workshop on Embedded networked sensors - EmNets '07*, 2007.

[88] "D4.5 – Second Software Integration and Preliminary Evaluation." GINSENG (INFSO-ICT-224282) Deliverable D4.5, WP4, 2011.

[89] J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels, T. Voigt, R. Sauter, and P. J. Marrón, "COOJA / MSPSim : Interoperability Testing for Wireless Sensor Networks Categories and Subject Descriptors," in *Proc. of the 2nd International Conference on Simulation Tools and Techniques, SIMUTools'09*, 2009.

[90] A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol." SICS Technical Report T2011:13, 2011.

[91] G. Zhou, J. A. Stankovic, and S. H. Son, "Crowded Spectrum in Wireless Sensor Networks," in *Proceedings of Third Workshop on Embedded Networked Sensors (EmNets)*, 2006.

[92] L. H. a. Correia, T.-D. Tran, V. N. S. S. Pereira, J. C. Giacomin, and J. M. Sá Silva, "DynMAC: A resistant MAC protocol to coexistence in wireless sensor networks," *Elsevier Comput. Networks*, vol. 76, pp. 1–16, 2015.

[93] P. Suriyachai, J. Brown, and U. Roedig, "Time-Critical Data Delivery in Wireless Sensor Networks," in *Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOSS10), Santa Barbara, USA*, 2010.

[94] S. Pollin, I. Tan, B. Hodge, C. Chun, and A. Bahai, "Harmful Coexistence Between 802.15.4 and 802.11: A Measurement-based Study," in *Proceeding of the International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CrownCom 2008)*, 2008.

[95] "D4.3 - Software integration (First evaluation phase)." GINSENG (INFSO-ICT-224282) Deliverable D4.3, WP4, 2010.

[96] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks," in *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, 2002.

[97] J. Zhao, R. Govindan, and D. Estrin, "Computing aggregates for monitoring wireless sensor networks," *Proc. First IEEE Int. Work. Sens. Netw. Protoc. Appl.*, pp. 139–148, 2003.

[98] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the sensor network debugger," *Proc. 3rd Int. Conf. Embed. networked Sens. Syst. - SenSys*

'05, p. 255, 2005.

[99] G. Tolle and D. Culler, "Design of an application-cooperative management system for wireless sensor networks," *Proceeedings Second Eur. Work. Wirel. Sens. Networks (EWSN 2005).*, pp. 121–132, 2005.

[100] S. Rost and H. Balakrishnan, "Memento : A Health Monitoring System for Wireless Sensor Networks," in *3rd Annual IEEE Conf. on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'06)*, 2006, pp. 575–584.

[101] M. Ringwald and R. Kay, "SNIF : A Comprehensive Tool for Passive Inspection of Sensor Networks," in *6. GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze,"* 2007, pp. 59–62.

[102] J. Beutel, M. Dyer, L. Meier, and L. Thiele, "Scalable topology control for deployment-support networks," in *4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*, 2005.

[103] M. Ringwald and K. Römer, "Monitoring and Debugging of Deployed Sensor Networks," in *2. GI/ITG KuVS Fachgespräch Systemsoftware für Pervasive Computing, Arbeitsberichte des Instituts für Informatik*, 2005.

[104] F. Dressler, R. Nebel, and A. Awad, "Distributed Passive Monitoring in Sensor Networks," in *26th IEEE Conference on Computer Communications (IEEE INFOCOM 2007), Demo Session*, 2007.

[105] A. Meier, M. Motani, H. Siquan, and S. Künzli, "DiMo : Distributed Node Monitoring in Wireless Sensor Networks," in *11th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2008)*, 2008.

[106] K. Liu, M. Li, M. Li, Z. Guo, Y. Liu, and F. Hong, "Passive diagnosis for wireless sensor networks," in *SenSys'08*, 2008, vol. 18, pp. 1132–1144.

[107] G. Yang and X. Hu, "Multi-Sensor Fusion," in *Body Sensor Networks*, Springer London, 2006, pp. 239–285.

[108] E. F. Nakamura, A. A. F. Loureiro, and A. C. Frery, "Information Fusion for Wireless Sensor Networks : Methods , Models , and Classifications," *ACM Comput. Surv.*, vol. 39, no. 3, 2007.

[109] F. E. WHITE, "Data fusion lexicon." U. S. Department of Defence, Subpanel of the Joint Directors of Laboratories, 1991.

[110] D. L. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proc. IEEE*, vol. 85, no. 1, pp. 6–23, 1997.

[111] W. Lucien, "SOME TERMS OF REFERENCE IN DATA FUSION," *IEEE Trans. Geosci. Remote Sens.*, pp. 1190–1193, 1999.

[112] A. Abdelgawad and M. Bayoumi, "Resource-Aware Data Fusion Algorithms for Wireless Sensor Networks," in *Resource-Aware Data Fusion Algorithms for Wireless Sensor Networks*, vol. 118, Boston, MA: Springer US, 2012.

[113] B. V Dasarathy, "Sensor Fusion Potential Exploitation — Innovative Architectures and Illustrative Applications," *Proc. IEEE*, vol. 85, no. 1, pp. 24–38, 1997.

[114] H. F. Durrant-Whyte, "Sensor models and multisensor integration," *Int. J. Rob. Res.*, vol. 7, no. 6, pp. 97–113, 1988.

[115] W. Elmenreich, "Sensor Fusion in Time-Triggered Systems," Ph.D Thesis, Vienna University of Technology, 2002.

[116] P. Mohanty, S. Panigrahi, N. Sarma, and S. S. Satapathy, "Security issues in Wireless Sensor Network data gathering protocols: a survey.," *J. Theor. Appl. Inf. Technol.*, 2010.

[117] R. Rajagopalan and P. K. Varshney, "Data aggregation techniques in sensor networks: A survey," *IEEE Commun. Surv. Tutorials*, vol. 8, no. 4, pp. 48–63, 2006.

[118] Y. Chen, J. Shu, S. Zhang, L. Liu, and L. Sun, "Data Fusion in Wireless Sensor Networks," in *2009 Second International Symposium on Electronic Commerce and Security*, 2009, pp. 504–509.

[119] Q. Wang, Y. Zhu, and L. Cheng, "Reprogramming wireless sensor networks: challenges and approaches," *IEEE Netw.*, vol. 20, no. 3, pp. 48–55, 2006.

Appendix

A

Appendix A – WMCAP details

In this section, additional details about the operation of the *WSNs Metrics Collection and Alert Protocol* (WMCAP) that uses fusion to collect metrics will be presented. Some selected details and schemes are repeated to clarify the explanation.

A.1. GENERAL ASSUMPTIONS

In order to enable the operation of the protocol some assumptions must be taken:

- Every node knows its neighbours, including which are child and which is the upstream node;
- Routing exists and is provided by other protocol;
- The protocol does not provide clock synchronization – to calculate metrics that require clock synchronization, an external synchronization protocol must be used;
- All the nodes must know the metrics identification codes and their respective methods of calculation and fusion functions - this information must be set before deployment or by using wireless nodes reprogramming.

A.2. BASE PROTOCOL AND EXTENDED VERSION

The proposed protocol has two versions, a base version and an extended version. The base version assumes that all metrics will be sent fused to upstream nodes and relies on alerts to inform directly the sink that a threshold has been surpassed. The extended protocol enables more flexibility, which is traded by less optimization. The extended version permits that each node may have different settings in what respects metrics to calculate and send, and metrics to fuse. Using this version, the network manager may set fusion rules that are only active in specified nodes and that only include performance data from those nodes. This data is fused and sent to the sink with a fusion rule number so that the sink may differentiate it from other packets. It may also enable dynamically setting of individual metrics that are sent by specific nodes. Mobility is also supported by the inclusion of specific messages to remove and add nodes from the neighbour list of a node. Next table presents the messages used by each version of the protocol.

Tables Table A-1 and Table A-2 show respectively the summary of the features for each version and the messages used.

TABLE A-1 – DIFFERENCES BETWEEN THE TWO VERSIONS OF THE PROTOCOL.

Base version
Allows for the dynamic setting of metrics and their healthy limits; Sink may query metrics from nodes; Activation and deactivation of nodes by the sink; Nodes send all metrics fused to upstream neighbours or do not send any metrics; Nodes send alerts directly to the sink if a metric threshold is crossed; Nodes send alerts directly to the sink if a 1-hop neighbour is presumed dead;
Extended version
Allows for the dynamic setting of metrics, their fusion options and rules, and their healthy limits; Sink may query metrics from nodes; Activation and deactivation of nodes by the sink; Nodes may send fused metrics to upstream nodes, individual metrics directly to the sink, or special fused metrics defined by a fusion rule (sent to other nodes that have the same rules); Nodes send alerts directly to the sink if both a metric and its number of sequential cumulative warnings threshold were crossed; Nodes send alerts directly to the sink if a 1-hop neighbour is presumed dead; New nodes that connect are reported to the sink; Nodes may be removed by the neighbour list; Update time is dynamic;

TABLE A-2 – MESSAGES USED BY THE TWO VERSIONS OF THE PROTOCOL

Messages used by both versions	**Messages specific to the extended version**
SET_NODES	METRIC_REPORT_SINGLE
METRIC_QUERY	METRIC_REPORT_FUSION_RULE
METRIC_QUERY_REPORT	ALERT_NEW_NODE
METRIC_REPORT_FUSION	REMOVE_NODE
ALERT_METRIC_REPORT_OUT_OF_BOUNDS	
ALERT_DISCONNECTION	
ACTIVATE_NODE	
IGNORE_NODE	

The reason for having two versions relies in the characteristics of WSNs. As the resources are scarce, every optimization that removes unnecessary overhead may lead to a longer lifetime of the node. The base protocol version has fewer messages and enables an easier protocol operation. It also allows for code and memory optimizations.

A.3. PROTOCOL CONSIDERATIONS

As the functions for metrics calculation and fusion must be already defined in nodes, their initial selection is of utmost importance. In spite of both versions being able to dynamically change the setup values during operation, or request the use of different metrics, the protocol does not provide by itself the means for their calculation. In the cases where changes to metrics exist or when fusion functions change, reprogramming during operation is necessary. When individual wired reprogramming is not possible, wireless reprogramming [119] may be the answer. In spite of causing a temporary overhead in the network, it provides a solution for sporadic upgrades or modifications of

the code running in each WSN node. In planned WSNs, the trade-off between additional functionalities and efficiency favours the latter.

A planned WSN with controlled performance should specify all the metrics to use and their healthy ranges before deployment, providing for code and resources optimization. In this scenario, the *SET_NODES* message is only used to alter healthy ranges of the already used metrics. In fact, as most of the WSNs with performance control are carefully planned, it is possible, at deployment time, to know most of its requirements. Also, in these networks, changes to initial settings do not occur frequently.

A.4. PROTOCOL MESSAGES

This section describes each of the specific packets used by the protocol.

Common header

SOURCE_ ID	DESTINATION _ID	SENDER_TIMESTAMP	NUMBER_OF _HOPS	PACKET_SEQUENCE_ NUMBER

Note: The protocol proposed does not imply synchronization between nodes, resulting in unreliable timestamps if operations are done involving timestamps from different nodes. However, clock synchronization is possible recurring to specific synchronization protocols. This is necessary if clock synchronization is essential for the calculation of specific metrics.

Packet type

The protocol defines 12 different messages for its operation (8 in the base version).

- **From sink to node:**
  - SET_NODES (S_N) – Protocol setup message that indicates the range of acceptable values for a metric, the metric update time and the alive threshold. It is sent to a specific node or to all nodes, on the start of the network or if the values are changed during operation.

TYPE	MAX_UP DATE_T IME	MIN_UP DATE_T IME	ALIVE_TH RESHOLD	N_METR ICS	METRIC _ID #1	FUSION #1	MAX_ME TRIC_V ALUE #1	MIN_ME TRIC_V ALUE #1	MAX_ WARN INGS #1	METRIC _ID #2	. . .
S_N	{secon ds}	{secon ds}	{seconds }	{numbe r}	{Metri c ID}	{no=0 | fusion _rule}	{ metric value}	{ metric value}	{ numb er}		

TYPE indicates the type of message; *MAX_UPDATE_TIME* and *MIN_UPDATE_TIME* specify the bounds for reporting activities from the nodes; *ALIVE_THRESHOLD* specifies the maximum time a contact from a 1-hop neighbour node can be missing, alerting for a probably dead node; *N_METRICS* indicates the number of metrics that are specified in this specific packet; For each

metric specified by its *METRIC_ID*, several parameters are set: *FUSION* that indicates if a specific metric should be sent fused or in individual values (if not fused the value is 0, if fused the value is the fusion rule number), *MAX_METRIC_VALUE* and *MIN_METRIC_VALUE* to limit the healthy bounds of a metric and the *MAX_WARNINGS* that specifies the number of cumulative warnings that are accepted for that metric (i.e., the number of out of bound values that are accepted before an alert message is sent to the sink).

Fusion rules and the possibility of sending individual metrics **only exist in the extended version** of the protocol.

The *SET_NODES* is confirmed by the reception of a *METRIC_QUERY_REPORT* with *QUERY_ID*=0.

- METRIC_QUERY (M_Q) – This message queries one of the performance metrics, of one or all nodes. The goal is to assess metric values before the network starts operating normally, or to retrieve specific metric values when in operation or debugging phases. It can either request a specific metric from one node. It only addresses one metric at each request and the response is always not fused.

TYPE	QUERY_ID	NODE_ID	METRIC_ID
M_Q	{Query ID}	{Node ID}	{Metric ID}

TYPE indicates the type of message. *QUERY_ID* references the identification of the query made by the sink, and is also used in the response, allowing the sink to easily identify it; *NODE_ID* references the node to which the metric referrers (e.g. distinguish the node referred by the delay hop metric sent by a node – each node may have multiple neighbours); *METRIC_ID* references the metric to retrieve.

- ACTIVATE_NODE (A_NODE) – Message that informs the network that a specific node is to be activated for performance collection. It is used when the control software in the sink informs nodes that they can start accepting performance values from that node. Following this message a SET_NODES message is sent to the specific node being activated, containing the metric boundaries to be applied.

TYPE	NODE_ID
A_NODE	{Node ID}

TYPE indicates the type of message. *NODE_ID* references the node to be activated.

- IGNORE_NODE (I_NODE) – Message that informs the network that a specific node is to be ignored for performance collection purposes. Is used when the

control software in the sink does not want performance values from a node or when wants to mark a specific node as compromised (either by an attack or an anomaly).

TYPE	NODE_ID
I_NODE	{Node ID}

TYPE indicates the type of message. *NODE_ID* references the node to be ignored.

- REMOVE_NODE (R_NODE) – Message that informs a specific node that a node is to be removed from its neighbour list. It is used when the control software informs a node that other node is no longer in its neighbour list and that for that reason it can cease sending alerts.

TYPE	NODE_ID
R_NODE	{Node ID}

TYPE indicates the type of message. *NODE_ID* references the node to be removed.

- **From node to sink:**
  - METRIC_QUERY_REPORT (M_QR) – This message reports a response to a specific query made by the sink using the METRIC_QUERY message. It only addresses one metric at each time and is never fused.

TYPE	QUERY_ID	METRIC_VALUE
M_QR	{Query ID}	{value}

TYPE indicates the type of message. QUERY_ID references the query that is being responded (previously received in a METRIC_QUERY packet) – the sink can easily interpret the received values by knowing which was the query that triggered the response. Therefore, the metric identification is not necessary.

  - METRIC_REPORT_SINGLE (MR_S) – This message reports one or more individual metrics from a node, i.e. metrics calculated in the node, which were not fused with others.

TYPE	N_METRICS	METRIC_ID #1	METRIC_VALUE #1	METRIC_ID #2	...
MR_S	{number}	{Metric ID}	{metric value}	{Metric ID}	

TYPE indicates the type of message. *N_METRICS* indicates the number of metrics that are specified in this specific packet - for each metric specified by its *METRIC_ID*, the corresponding metric value is sent. Only metrics that are calculated in the node and refer only to the node are sent. Metrics calculated in

the node but referring to other nodes (e.g. delay hop from neighbours) are not sent as would require a *NODE_ID* to be specified.

- METRIC_REPORT_FUSION (MR_F) – This message reports fused metrics and is sent to the upstream node towards the sink. The number of samples used in calculations and the fusion levels must be indicated. The number of samples indicates the number of node metric samples that were used to obtain the metric transmitted. The fusion level indicates the number of fusion processes that have been done to the data.

TYPE	N_ METRICS	METRIC_ ID #1	METRIC_ VALUE #1	N_ SAMPLES #1	FUSION_ LEVELS #1	METRIC_ ID #2	...
MR_F	{number }	{Metric ID}	{metric value}	{ sample}	{levels }	{Metric ID}	

TYPE indicates the type of message. *N_METRICS* indicates the number of metrics that are specified in this specific packet. For each metric specified by its *METRIC_ID*, a corresponding metric value is sent, together with information of the number of samples used and the number of fusion operations done to the data.

- METRIC_REPORT_FUSION_RULES (MR_FR) – This message is used when nodes fusion depends on specific rules and where different nodes may have different fusion rules. Only one metric is sent in the same packet.

TYPE	FUSION_RULE	N_SAMPLES	FUSION_LEVELS
MR_FR	{fusion rule}	{samples}	{levels}

TYPE indicates the type of message. The number of samples used in calculations and the fusion level are also sent. The number of samples indicates the number of node metric samples that were used to obtain the metric transmitted. The fusion level indicates the number of fusion processes that have been done to the data. No metric is specified as each fusion rule has a specific metric (the same metric may have different fusion rules that apply to different nodes to enable different number of fusion aggregates in the network).

- ALERT_METRIC_REPORT_OUT_OF_BOUNDS (A_MROOB) – Message that informs upstream nodes that an out-of-bounds value for a metric has been detected in a specific node. This message should be sent immediately and directly to the sink without any fusion.

TYPE	NODE_ID	METRIC_ID	METRIC_VALUE
A_MROOB	{Node ID}	{Metric ID}	{metric value}

TYPE indicates the type of message. *NODE_ID* specifies the node to which the metric refers. It can be the same node that sends the message (if the metric is calculated referring to the same node) or other node (e.g. delay-hop is calculated

in a node but refers to the values obtained from the previous hop node). *METRIC_ID* specifies the metric calculated and METRIC_VALUE its corresponding value.

- ALERT_DISCONNECTION (A_D) – Message that informs the sink that a specific node did not respond or sent a message in the predefined time limit (defined in a SET_NODES message). It indicates that the node may have a problem. This message is not fused and is immediately forwarded to sink.

TYPE	NODE_ID
A_D	{Node ID}

TYPE indicates the type of message. *NODE_ID* specifies the node that did not contact in a predefined time.

- ALERT_NEW_NODE (A_NN) – Message that informs the sink that there is a new node in the network. This message is sent to the sink by the node to which the new node attaches. This message is not fused and is immediately forwarded to sink.

TYPE	NODE_ID
A_NN	{New Node ID}

A.5. MANAGEMENT INFORMATION BASE (MIB)

In order to enable protocol operation some values must be saved in nodes internal memory.

- For each metric reported by the node:
  - FUSION – indicates if the metric is to be fused before being forwarded in the network;
  - MAX_METRIC_VALUE – maximum healthy value of a metric;
  - MIN_METRIC_VALUE – minimum healthy value a metric value;
  - MAX_WARNINGS – number of cumulative warnings any metric may have.
- General operating values:
  - MAX_UPDATE_TIME – maximum time a node can wait to send a performance update to the next upstream node;
  - MIN_UPDATE_TIME – minimum time a node can wait to send a performance update to the next upstream node
  - UPDATE_TIME = [MIN_UPDATE_TIME, MAX_UPDATE_TIME] – current update time for the node;
  - ALIVE_THRESHOLD – indicates the maximum time a node can wait for an update of its direct neighbours

- Neighbour list
  - Status table having one line per neighbour/metric - for each (downstream) neighbour:
    - □ Indicates if the neighbour node is to be ignored;
    - □ Saves time/date of last info received from each neighbour (only the last info, history is possible but not advisable due to memory restrictions);
    - □ Saves values of each metrics received from neighbours and, for each one, a field informing how many samples constitute the value (i.e from how many sources it comes from) and the number of times it has been fused before. It also has a warning field for each node, stating if warnings have been fired for that node in that metric.

A.6. PROTOCOL OPERATION

This sub-section addresses the protocol operations by phase of the WSNs life cycle.

A.6.1. Protocol initiation (network deployment)

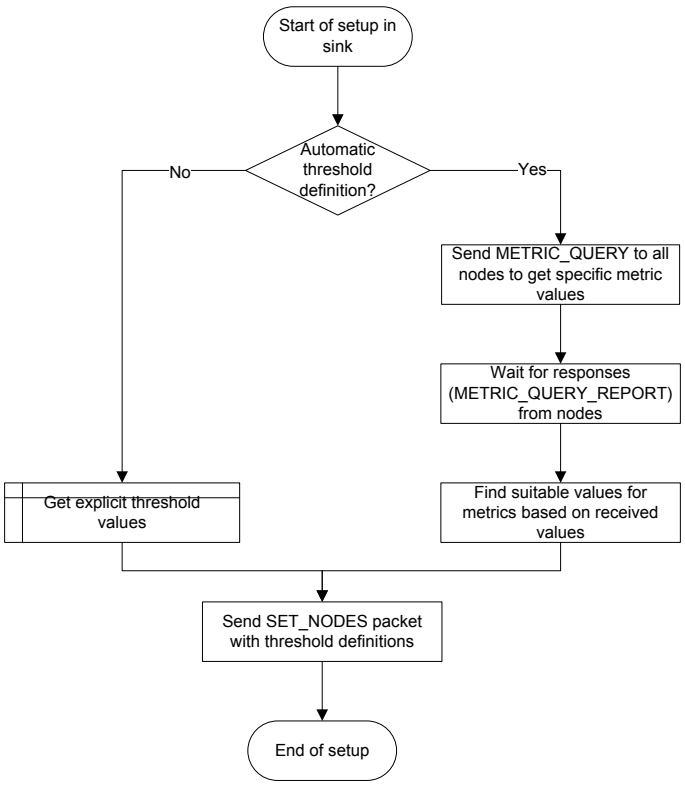


Fig. A-1 – WMCAP: initialization.

During this phase the thresholds for each metric are set. This can be made automatically by retrieving metrics from nodes while the network is running in the best conditions possible, or can be set explicitly by the administrator based on the used applications

performance requirements (Fig. A-1). Although the process described is focused in the deployment phase, *SET_NODES* messages can be used at any time to change the initial settings.

Initial threshold definition

This step is made when installing the network or during its initial tests; if necessary, the network administrator can explicitly change all thresholds at any time.

- Metric threshold calculation
  - Automatic: Thresholds are calculated in the sink using the data received from nodes.
  - Explicit: Network manager explicitly defines thresholds that correspond to the network QoSensing requirements.
- Automatic metric threshold definition (done in controlled environments when the network is in its optimal state):
  - Sink sends a METRIC_QUERY message to every node, identifying the specific metric for the nodes to report. The message is sent to all nodes in the network with a specific QUERY_ID field.
  - This message demands a response using a METRIC_QUERY_ REPORT message, with the same QUERY_ID field value, which identifies the query. If a response is not received from a node, the query message is repeated for the nodes from where response did not came.
  - If after 3 attempts there is no response of the node, an IGNORE_NODE message, with NODE_ID field set to the node that did not respond, is sent. If later a packet is received from that node, an ACTIVATE_NODE message with the same NODE_ID is sent to all nodes.

Thresholds propagation

- Sink sends a SET_NODES message to all nodes:
  - It defines the limits for every metric previously defined (MAX_METRIC_VALUE, MIN_METRIC_VALUE, MAX_WARNINGS); MAX_WARNINGS states how many warnings may be triggered for a metric without sending an alert to the sink;
  - The packet also contains a maximum and minimum time for the updating of metrics (MAX_UPDATE_TIME, MIN_UPDATE_TIME);
  - It sets if the data for that metrics is to be fused (FUSION);
  - It also sends an ALIVE_THRESHOLD, defining a threshold for assuming that a neighbour is dead; If a node that previously received a METRIC_QUERY packet with QUERY_ID=0, does not receive a SET_NODES packet in a specified time, it repeats the METRIC_QUERY_REPORT packet;
- The value of MAX_UPDATE_TIME < ALIVE_THRESHOLD;

- The nodes initially assume UPDATE_TIME = MIN_UPDATE_TIME;
- The SET_NODES message is acknowledge by a node using a METRIC_QUERY_REPORT with QUERY_ID=0.
  - If the sink receives no response from nodes, other 2 attempts are used, sending a SET_NODES message to the specific nodes that did not respond.
  - If after 3 attempts there is no response of the node, an IGNORE_NODE message, with NODE_ID field set to the node that did not respond, is sent to all nodes. If later a packet is received from that node, an ACTIVATE_NODE message with the same NODE_ID is sent to all nodes. After this initial phase the network is set and starts its operation phase.

A.6.2. Operation phase (network operation)

During this phase common nodes (nodes that are not the sink) receive and send performance packets from upstream and downstream nodes.

A.6.2.1. Reception of packets

Depending on the source of the received performance packet, different operations are executed. The initial selection process is shown in the flowchart depicted in Fig. A-2.

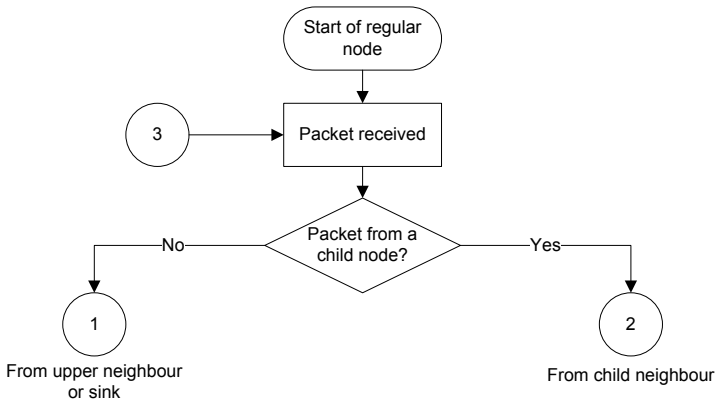


Fig. A-2 – WMCAP: reception of performance messages in a common node (1, 2 and 3 continue in Fig. A-3 and Fig. A-4).

Packet received from upstream neighbour or sink (parent node)

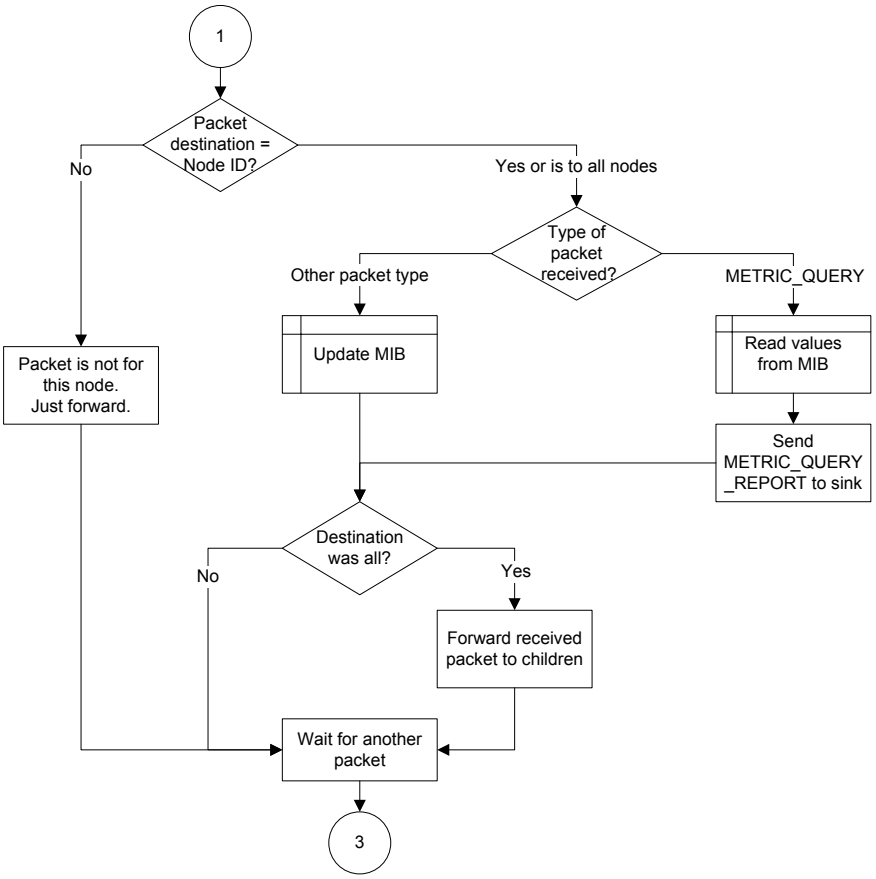


Fig. A-3 - WMCAP: reception of packets from upstream nodes.

On receiving a packet from an upstream node the first operation is to check its destination:

- If the destination is not for the current node, the packet is just forwarded;
- If the packet is for the current node or to all nodes:
  - If the packet received is a METRIC_QUERY:
    - The metric request is calculated and send to sink using a METRIC_QUERY_REPORT, with the field QUERY_ID being the same that was received, providing an identification of the query to the sink;
  - If the packet received is a SET_NODES:
    - Each node sets its values according to the message received and replies to sink with a METRIC_QUERY_REPORT with QUERY_ID field set to 0;
  - If the packet received is an ACTIVATE_NODE, IGNORE_NODE or REMOVE_NODE:
    - The node activates a neighbour node, ignores it, or removes it from its MIB.
  - If the destination was 'all nodes' the original message is also forwarded to child nodes.

Packet received from downstream neighbour

The operations executed when receiving a performance packet from a downstream neighbour depend on the version of the protocol used (Fig. A-4 and Fig. A-5).

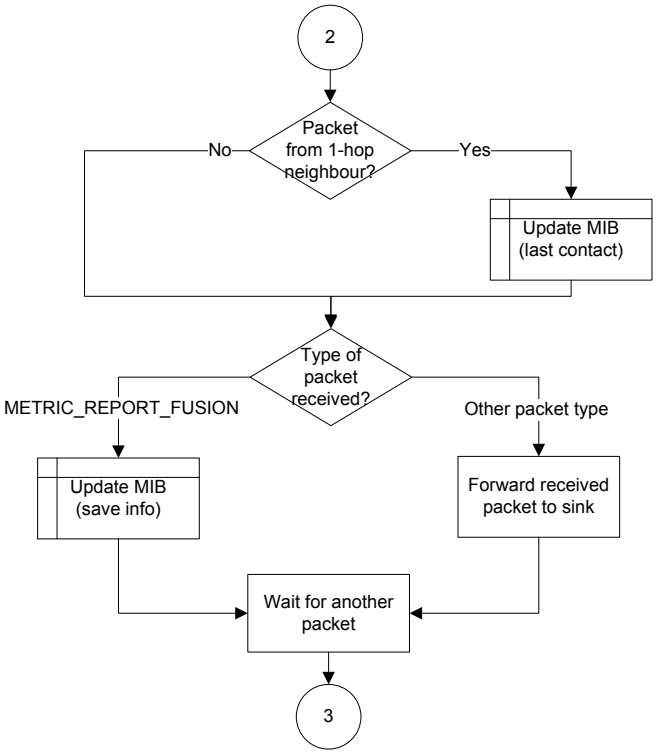


Fig. A-4 - WMCAP: reception of packets from downstream nodes – base version.

In the base version, when a node receives a packet from a downstream neighbour the first thing is detect if it is a 1-hop neighbour. If it is, then the local MIB is updated with the time of the arrival, so that the time of last contact from that specific node is saved. By using local time, the protocol does not obligate, by itself, time synchronization between nodes. It only saves the last time of contact from each node – history was possible but not advisable due to memory restrictions. Then, the operation depends on the type of packet received:

- If the packet received is of type METRIC_REPORT_FUSION, the local MIB is updated with the metrics received from the neighbour node:
  - The data is saved in a line that corresponds to the neighbour node from which the packet was sent. If the MIB contains data from the same node that has not yet been sent, it is overwritten, saving the more recent data.
- If the packet received is of other type, it is just forwarded to the sink.

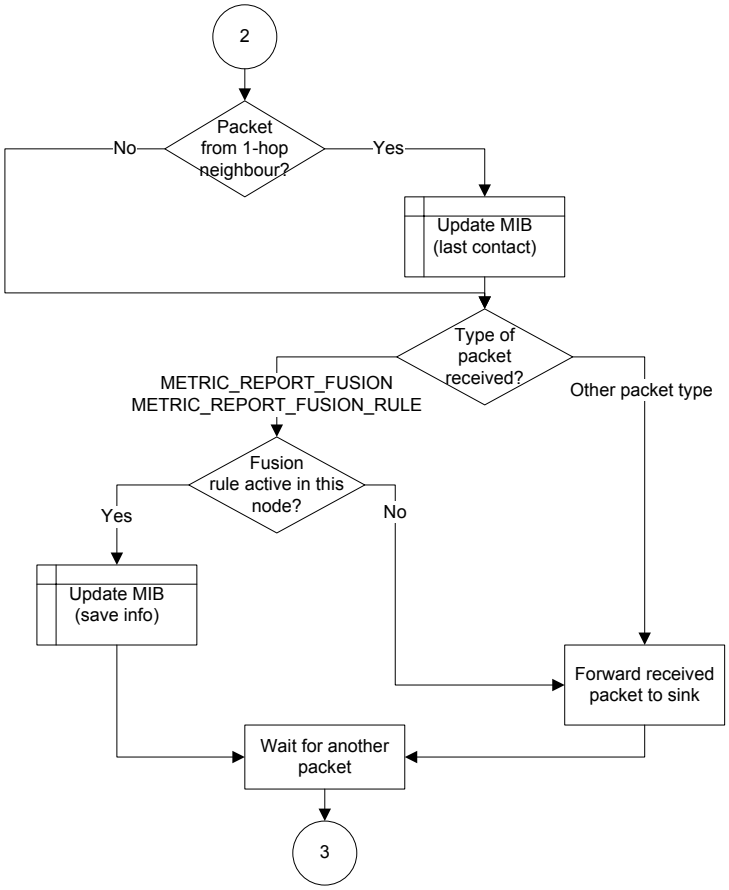


Fig. A-5 - WMCAP: reception of packets from downstream nodes – extended version.

In the extended version, the differences are that both *METRIC_REPORT_FUSION* and *METRIC_REPORT_FUSION_RULE* messages are processed. If the field *FUSION_RULE* of the *METRIC_REPORT_FUSION_RULE* is not defined in the current node, that message is just forwarded.

A.6.2.2. Internal node reporting operations

From time to time, defined by UPDATE_TIME, each node calculates its performance metrics and sends the results to the sink (Fig. A-6 and Fig. A-7).

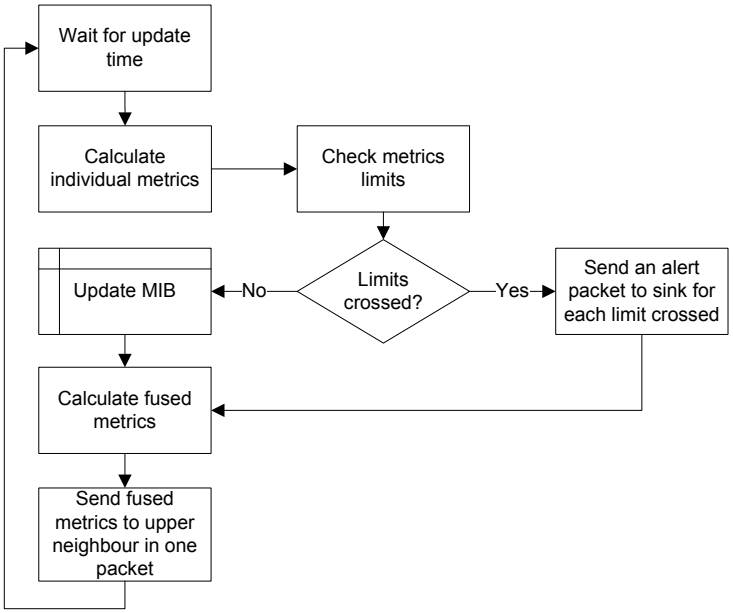


Fig. A-6 – WMCAP: metrics update report – base version.

In the base version, at each update interval, the current node calculates its metrics. The metrics can be calculated from data of the node, or considering data brought by previously received packets (as an example, to calculate the 1-hop delay from its neighbours, it uses data obtained from the packets received from them). Then, each metric value is checked according to predefined limits, to see if its value is between healthy values. If not, it sends an *ALERT_METRIC_REPORT_OUT_OF_BOUNDS* message to the sink, with the *METRIC_ID* and the *METRIC_VALUE* of the calculated metric. If it is necessary to refer the node to which it relates (in the situations where the metric is calculated with values obtained from other nodes – e.g. 1-hop delay) *NODE_ID* can be set to the appropriate value. It also checks the last time of contact of each 1-hop neighbour node. If any of those contacts happened too long ago (>*ALIVE_THRESHOLD*) an *ALERT_DISCONNECTION* alert message is sent to the sink.

Finally, all local metrics within limits (i.e. excluding outliers) are fused with the metrics received from neighbours. The number of samples that existed in each of the previously received packets is added with the number of samples that the node itself produced (if in-node aggregation is being done a metric calculated in a node may have more than one sample). The number of fusion operations is also added and increased by 1 (to add the present fusion). After all operations, a *METRIC_REPORT_FUSION* packet is sent to the upstream node.

After each update round, the metrics data in the MIB is set as dirty, meaning that the values were already sent.

In this protocol no reliability is assumed. If the packet sent with the information is lost there will be no way to retransmit the previous performance data. If a higher-level node

detects the absence of response that exceeds the predefined time (*ALIVE_THRESHOLD*), an alert message will be sent to the sink respecting this node.

Update time is dynamic, being adaptable to the current observed QoSensing of the network. If there were no outliers in the previous update round, if all nodes sent info, and if there were no alerts forwarded from downstream level nodes, then *UPDATE_TI*ME = *MAX_UPDATE_TIME*, else *UPDATE_TIME = MIN_UPDATE_TIME*.

While possible, the proposed protocol does not permit the shutdown of specific alert messages, nor puts a limit to their number. As an example, if a node's energy falls behind a predefined threshold, it will keep sending alerts till the problem is solved. To stop that behaviour three options are available: the problem is solved, a new *SET_NODES* message with different metric limits is sent to the node, or an *IGNORE_NODE* is sent by the sink to inform nodes that they should ignore the messages from that specific node.

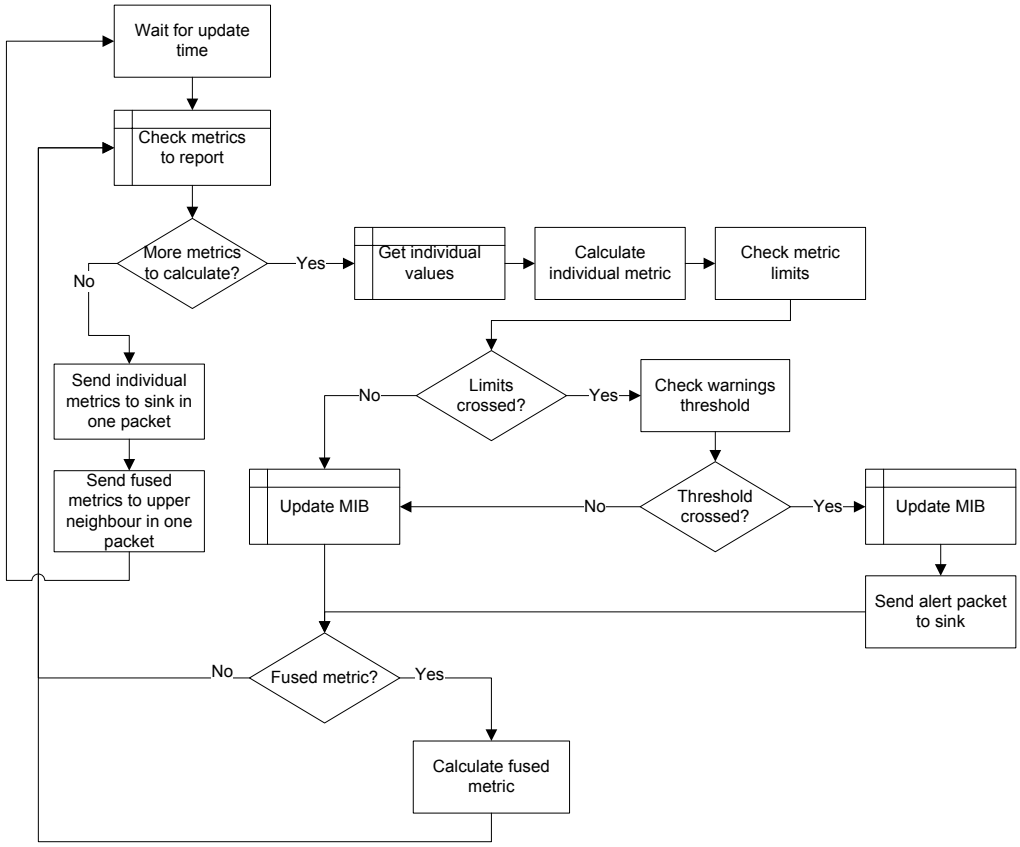


Fig. A-7 - WMCAP: metrics update report – extended version.

In the extended version, some new features are possible:

- Both fused metrics as individual metrics may be sent;
  - Individual metrics are sent in *METRIC_REPORT_SINGLE* packets while general fused metrics use *METRIC_REPORT_FUSION* packets;

- Specific metrics with specific rules also send *METRIC_REPORT_FUSION_RULE* packets; These last packets allow for the existence of selected fused data from specific node sets (the same metric in a node may belong to different rules);
- A warning threshold, defined in the field *MAX_WARNINGS* of the *SET_NODES* message, can be set for each individual metric calculated in the node. The number of warnings is increased for each sequential metric value out of healthy bounds. An *ALERT_METRIC_REPORT_OUT_OF_BOUNDS* packet is sent only if this threshold is exceeded. Otherwise the value out-of-bounds is just ignored. On receiving a healthy metric from a node, the number of warnings for that specific metric is reset.

A.6.3. Protocol debug (network debug&recovery)

During this phase the sink sends at least one *METRIC_QUERY* message to specific nodes, inquiring about a specific metric. The nodes answer by sending a *METRIC_QUERY_REPORT* message with the *QUERY_ID* field set to the one received.

A.6.4. Dealing with changes in the topology

Changes in the topology may occur from 3 different reasons: a node can be added to the network, a node can be removed, a node can change place. Despite the cause, a change in the topology affects the proposed protocol in that the aggregates contain values that are different from the previous ones, making it more difficult to compare values and misleading the network manager.

To deal with the topology changes the protocol has the following features:

Option 1: Assume a static deployment where nodes do not change place and the routing is static.

The neighbour list is pre-loaded in each of the node's MIB and the protocol uses packet types *IGNORE_NODE* and *ACTIVATE_NODE* to manage the inclusion or exclusion of a specific node (that must exist initially, i.e., nodes are not new to the network).

Option 2: Rely on the routing protocol to update the routing table of the nodes and give that information to the sink.

The underlying routing protocol changes each node's routing table, each node's MIB and informs the sink of the changes. No further messages are necessary.

Option 3: Use the protocol packet types to address the issue.

In case of Option 3 the protocol must address the changes in topology notifying the sink that a change occurred. The 3 different reasons to the change of topology, stated before, lead to 2 different responses.

A new node is added to the network or an existing node changes place:

- On receiving a message from a node that is not included in its neighbour list, a node sends an *ALERT_NEW_NODE* packet informing the sink that a new node was added (it is not necessary for other nodes to acknowledge that fact as each node is only responsible for its neighbours). On receiving that message, the sink sends a *REMOVE_NODE* packet to the node where the node was previously attached (if any), which implies its removal from its neighbour list), so that it can cease sending alerts. This implies that the sink is aware of the WSN topology.
- If the node is a new node, then an *ACTIVATE_NODE* must be sent to the node that sent the *ALERT_NEW_NODE* by the sink.

A node is removed from the network:

- This case is the same that happens when a node ceases transmitting. The node is simply deactivated. It is only removed if it connects to other node in the network.

In the proposed mechanism security questions are not considered, as they must be treated in a different level. So, it is assumed that a node only accepts packets from another if it has permission to do so.

A.7. SOME CONSIDERATIONS ABOUT THE PROTOCOL

Assuming a network with a sink and three levels of nodes, each having 2 child nodes, a normal performance reporting would need 12 messages in Level 3, 18 messages in Level 2 and 21 messages in Level 1. This corresponds to the performance packets generated by each node and to the additional packets that must be forwarded from child nodes. This number does not include any specific query messages from sink, nor additional packet with alerts. In the same scenario, using our proposed fusion protocol and considering that the data from metrics could be fused, 12 messages would be needed in Level 3, 6 messages in Level 2 and 1 message in Level 1. If metrics could not be fused, the number of messages was the same as in a normal performance reporting. The time wasted to fuse values is short as only simple calculations are made and is done while the node is not transmitting.

By lowering the number of messages needed to collect the network performance, interferences in communications between nodes are minimized, bandwidth is saved and energy is saved, while maintaining an effective performance surveillance that can even be made with a higher frequency.

A.8. THEORETICAL NUMBER OF PACKETS IN A WSN

The number of packets generated is one of the main reasons for the overhead implied in a WSN with controlled performance, wasting valuable bandwidth and contributing to the

increase of the number of interferences in the network. To have an idea of the expected impact of packets generated, some theoretical overhead estimation for the monitoring of WSNs was made. This estimation can be deduced by analysing the delivery path of the performance messages transmitted.

WSN used for monitoring can assume a variety of different topologies, with a different number of nodes to monitor. Some networks only monitor leaf nodes, while others can monitor all nodes. While each network may have more than one sink, each specific node usually reports to just one, being the other used only if there is a problem with the connection to the first. So, it can be assumed for calculation purposes that each network only reports to an individual sink. Also, it is assumed that packets have predefined and stable routes to the sink.

Considering the use of a continuous data delivery model for monitoring, i.e., the delivery of performance data every specific time interval, in each monitoring cycle, and being *D* the set of nodes in the network that deliver performance data, the number of performance packets delivered is presented in next table.

TABLE A-3 – CALCULATION OF THE NUMBER OF PACKETS CREATED IN ANY NETWORK.

Generic network	Without fusion	$\sum_{x=1}^{n} hops\_to\_sink_x, x \in D, n = \#D$ D = network nodes that produce data #D = number of nodes in the set	The total number of packets in the network equals the sum of the number of hops between each data producing node and the sink.
	With fusion	$\sum number\ of\ links$	The total number of packets in the network equals the sum of the number of links that participate in the delivering of data to the sink. The number is the same if only leaf nodes produce data and intermediate nodes forward it, and if all nodes produce data.

Nodes that produce data are nodes that produce monitoring data. Other nodes do nothing or just forward the existing packets. In these values, eventual packets of alerts are not considered.

Example: In the example below (Fig. A-8), without using fusion, 30 packets are generated from the initial 8 created at each reporting cycle. With fusion the number is reduced to 14 as only one packet travels to the sink in each hop.

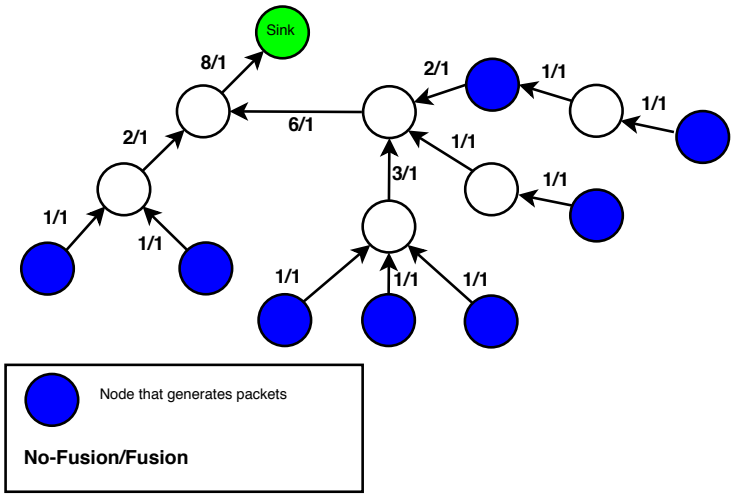


Fig. A-8 – Comparison between number of packets in a network with and without fusion.

In the case of having a tree structure where each node has the same number of sons as its father, and where all branches have the same height, calculations are facilitated. For this kind of tree, with a height *H* (*H* = number of levels-1) and with a number of sons from each node of *K*, the expected number of packets is as follows:

TABLE A-4 – CALCULATION OF THE NUMBER OF PACKETS CREATED IN A NETWORK WITH A TREE TOPOLOGY WITH ALL BRANCHES OF THE SAME HEIGHT AND SAME NUMBER OF CHILDREN

Number of packets generated in a network with a topology of a balanced tree with same height on every branch	Without Fusion	Only leafs generate packets	$k^{levels-1} * (levels - 1)$ *k* = number of child per node; *levels* = height of the tree+1;
		All nodes generate packets	$\sum_{n=2}^{levels} k^{n-1} * (n-1)$ *k* = number of child per node; *levels* = height of the tree+1;
	With Fusion	Independent of the number of nodes generating packets	$\sum_{n=2}^{levels} k^{n-1}$ *k* = number of child per node; *levels* = height of the tree+1;

In all types of network, the number of packets that reaches sink in each monitoring cycle is shown in Table A-5.

TABLE A-5 – NUMBER OF PACKETS THAT REACH THE SINK

Number of packets that reach the sink in each monitoring cycle, using any topology	Without fusion	Equals the number of nodes that generate packets
	With fusion	Equals the number of nodes 1-hop away from the sink (the number of sink children)