Un middleware fiable para el desarrollo de aplicaciones sobre redes inalámbricas de sensores y actores

(A Dependable Middleware for the Development of Applications for Wireless Sensor and Actor Networks)

PhD Thesis of:
Jaime H. Chen Gallardo

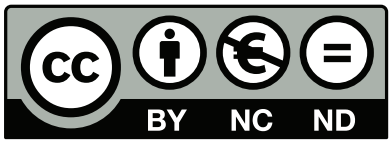
Advisors:
Dr. Bartolomé Rubio Muñoz
Dr. José M. Troya Linero

To obtain the degree of:
Doctor Ingeniero en Informática
Dept. Lenguajes y Ciencias de la Computación
Universidad de Málaga

UNIVERSIDAD
DE MÁLAGA

AUTOR: Jaime Hing Fong Chen Gallardo

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga



D. Bartolomé Rubio Muñoz, Profesor Titular de Universidad del Departamento de Lenguajes y Ciencias de la Computación de la E.T.S. de Ingeniería Informática de la Universidad de Málaga, y D. José M. Troya Linero, Catedrático de Universidad del Departamento de Lenguajes y Ciencias de la Computación de la E.T.S. de Ingeniería Informática de la Universidad de Málaga

Certifican

Que D. Jaime Hing Fong Chen Gallardo, Ingeniero en Informática, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo nuestra dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulada

Un middleware fiable para el desarrollo de aplicaciones sobre redes inalámbricas de sensores y actores
(A Dependable Middleware for the Development of Applications for Wireless Sensor and Actor Networks)

Revisado el presente trabajo, estimamos que puede ser presentado al tribunal que ha de juzgarlo, y autorizamos la presentación de esta Tesis Doctoral en la Universidad de Málaga.

Málaga, 10 de Enero de 2014

Fdo.: Bartolomé Rubio Muñoz
Titular de Universidad del Departamento
de Lenguajes y Ciencias de la Computación.

Fdo.: José M. Troya Linero
Catedrático de Universidad del Departamento
de Lenguajes y Ciencias de la Computación.

List of publications

During the course of writing this thesis the following publications have been produced (listed by category and in reverse chronological order):

Journal papers related to this thesis

- E. Cañete, J. Chen, M. Díaz, B. Rubio, J. M. Troya. Performance Analysis of Sensor Networks and Priority Queueing Systems. *Journal of Network and Computer Applications.* 2013. Elsevier. *In review*.

- A. Grilo, J. Chen, M. Díaz, D. Garrido and A. Casaca. An Integrated WSAN and SCADA System for Monitoring a Critical Infrastructure. *IEEE Transactions on Industrial Informatics* 2013, IEEE. *In review*.

- J. Chen, M. Díaz, B. Rubio, J. M. Troya. PS-QUASAR: A publish/subscribe QoS Aware Middleware for Wireless Sensor and Actor Networks. *Journal of Systems and Software.* Volume 86 Issue 6, June 2013, pages 1650-1662. Elsevier. ISSN: 0164-1212. (JCR 2012 Impact Factor: 1.135 5-Year Impact Factor: 1.322)

Category Name	Total Journals in Category	Journal Rank in Category	Quartile in Category
COMPUTER SCIENCE, SOFTWARE ENGINEERING	105	41	Q2
COMPUTER SCIENCE, THEORY & METHODS	100	30	Q2

- J. Chen, M. Díaz, L. Llopis, B. Rubio and J. M. Troya. A Survey on quality of service support in wireless sensor and actor networks: requirements and challenges in the context of critical infrastructure protection. *Journal of Network and Computer*

Applications. Volume 34 Issue 4, July 2011, pages 1225-1239. Elsevier. ISSN: 1084-8045. (JCR 2011 Impact Factor: 1.065 5-Year Impact Factor: 0.960).

Category Name	Total Journals in Category	Journal Rank in Category	Quartile in Category
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	50	21	Q2
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	99	62	Q3
COMPUTER SCIENCE, SOFTWARE ENGINEERING	104	36	Q2

Conferences related to this thesis

- E. Cañete, J. Chen, M. Díaz, B. Rubio and J. M. Troya. Performance Analysis of a Sensor Lineal Network and Priority Queueing Systems: Network Layer Vs. Data Link Layer. *In Proc. of Simposio de Sistemas de Tiempo Real, CEDI 2013* Madrid, Spain, 18-19 September 2013. pages 57-62. (Acceptance Rate: -).

- J. Chen, M. Díaz, B. Rubio and J. M. Troya. RAISE: RAIlway infrastructure health monitoring using Wireless SEnsor Networks. *In Proc. of S-CUBE 4th International Conference on Sensor Systems and Software* Lucca, Italy, 11-12 June 2013. pages 1-15. (Acceptance Rate: 50%).

Other related journal papers

- E. Cañete, J. Chen, R. M. Luque and B. Rubio. NeuralSens: A Neural Network based Framework to Allow Dynamic Adaptation in Wireless Sensor and Actor Networks. *Journal of Network and Computer Applications.* Volume 35 Issue 1, January 2012, pages 382-393. Elservier. (JCR 2012 Impact Factor: 1.467. 5-Year Impact Factor: 1.251).

Category Name	Total Journals in Category	Journal Rank in Category	Quartile in Category
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	50	11	Q1
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	99	44	Q2
COMPUTER SCIENCE, SOFTWARE ENGINEERING	105	24	Q1

- E. Cañete, J. Chen, M. Díaz, L. Llopis and B. Rubio. A service-oriented approach to facilitate WSAN application development. *Ad Hoc Networks.* Volume 9 Issue 3, May 2011, pages 430-452. Elsevier. (JCR 2011 Impact Factor: 2.110. 5-Year Impact Factor:-).

Category Name	Total Journals in Category	Journal Rank in Category	Quartile in Category
COMPUTER SCIENCE, INFORMATION SYSTEMS	135	20	Q1
TELECOMMUNICATIONS	79	13	Q1

Other related conferences

- E. Calderón, J. Chen and B. Rubio. DDS4RIS: a DDS implementation for wireless sensor networks. *In Proc. of 2010 International conference on computer and information technology.* Dubai, UAE, 25-27 January 2011. pages 15-19. World Academy of Science, Engineering and Technology, 2010. ISSN:1307-6892
  (Acceptance Rate: -).

- E. Cañete, J. Chen, L. Llopis, R. Marcos Luque and B. Rubio. Dynamic adaptation in Wireless Sensor Networks using Neural Networks. *In Hans Weghom and Pedro Isaías (eds.). IADIS International Conference, Applied Computing.* Rome, Italy, 19-21 November 2009. pages 3-7. IADIS Press, 2009. ISBN: 978-972-8924-97-3.
  (Acceptance Rate: 20%).

- E. Cañete, J. Chen, L. Llopis, R. Marcos Luque and B. Rubio. Case study: tractor safety; avoiding overturns and farmland monitoring using WSANs and the Useme Framework. *In Hans Weghom and Pedro Isaías (eds.). IADIS International Conference, Applied Computing.* Rome, Italy, 19-21 November 2009. pages 202-206. IADIS Press, 2009. ISBN: 978-972-8924-97-3.
  (Acceptance Rate: 20%).

- E. Cañete, J. Chen, M. Díaz, L. Llopis and B. Rubio. A Service-Oriented middleware for wireless sensor and actor networks. *In Sixth International Conference on Information Technology: new generations.* Las Vegas, USA, 27-29 April 2009. pages 575-580. IEEE Computer Society: Los Alamitos, CA.
  (Acceptance Rate: 29%).

- E. Cañete; J. Chen; M. Díaz; L. Llopis y B. Rubio. USEME: a service-oriented framework for wireless sensor and actor networks. *In Klaus David...[et al], (eds.). Eighth International Workshop on Applications and Services in Wireless Networks.* ASWN 2008. 9-10 October 2008, Kassel, Germany. pages 47-53. IEEE Computer

Society: Los Alamitos, CA, 2008. ISBN: 978-0-7695-3389-6
(Acceptance Rate: -).

Contents

List of Tables

List of Figures

1

Introduction

Wireless Sensor Networks (WSNs) [1] have introduced an innovative way of monitoring and interacting with the environment. WSNs are networks composed of small embedded devices, called motes (Figure 1.1). Motes are self-powered devices that can sense the environment using onboard sensors and are able to communicate with each other by means of wireless transceivers and when deployed in large scenarios they form self-organizing networks that can effectively monitor large areas. The lack of wiring and their self-powered nature makes them really flexible. Moreover, the use of a large number of these devices introduces node redundancy which can be used to provide a high fault tolerance. This technology allows information to be sensed from the environment and transformed into digital data that can be stored and analyzed. Normal sensor devices are passive elements in the sense that they can only perceive the environment. However, applications can make use of actors, namely, resource rich devices that are capable not only of perceiving the environment but also of acting on it. These types of networks which include actors and sensors are called Wireless Sensor and Actors Networks (WSANs) [2].

Figure 1.1: Wireless sensors from different vendors

Early research to this technology started around 1980 with the Distributed Sensor Networks program at DARPA [3]. The advances in different technologies such as miniaturization processes, wireless, sensoring and microprocessors gave birth to the sensor devices. As time has gone on, we have witnessed improvements in the size of the devices, data transfer speed, available sensors, battery autonomy, etc. This has led the scientific community to propose and develop new applications where this technology can actually be applied. In fact WSANs are expected, by some, to be one of the most promising technologies in the near future. For example, according to Research and Markets WSANs are expected to become one of the top ten technologies that will have a big impact on the world over the next few years [4]. WSANs are expected to become part of what is called The Internet of Things [5], which is a huge network composed of a wide variety of electronic devices such as PCs, electrical appliances, mobile phones, embedded devices, sensors, etc (Figure 1.2). WSANs will be the senses of a huge nervous system that allows information to be felt, stored and analyzed.

There are a great number of applications where WSANs can be applied and their advantages have long been acknowledged in the research community. Their main application domain is the monitoring and controlling (by means of actors) of large scenarios. Applications where WSANs have been applied include pollution detection [6][7], agriculture [8][9][10], volcano and glacier monitoring [11][12], health monitoring [13][14], structural health monitoring [15][16], intrusion detection [17], motion tracking [18], among many others.

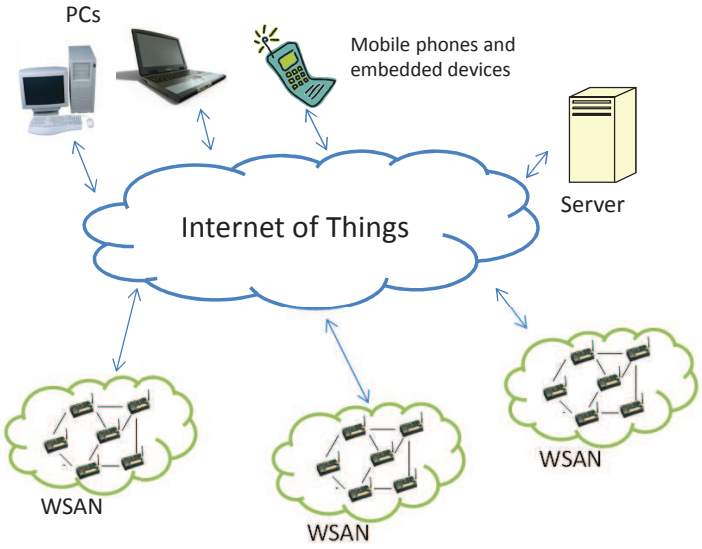


Figure 1.2: Internet of Things

1.1 Motivation

In spite of all that has been said in the introduction and despite being heralded as one of the most important technologies for the 21st century, WSANs have not yet become ubiquitous. It has been more than 30 years since the first research took place but, until now, the number of real applications (long-term deployments) where this technology has been applied is relatively low. What is more, according to Gartner [19], WSAN technology is at least 10 years away from becoming a mainstream technology. What has hampered the transition of WSANs from being a promising technology to being a fully-integrated technology in our daily lives? Some of the factors that may have hindered their development are the unreliable nature of wireless communications and the limited resources of the devices. Moreover, WSANs constitute a distributed system and as such all the difficulties of these kinds of systems equally apply to them. Furthermore, developing WSAN applications is not an easy task and intense research has been carried out to find new frameworks, tools, and middleware that provide higher levels of abstraction in order to simplify the developers' task [20][21]. In addition, new applications for WSANs demand new requirements and features from the sensor devices. For example, in recent years the possibilities of WSANs have been acknowledged as promising for the Critical Infrastructure Protection (CIP) field [22].

In this regard, WSANs have the potential to become an integral part of the protection of CIs such as electricity generation and transmission facilities, telecommunication systems, water supply, etc. Their distributed nature makes them particularly suitable against failures and attacks as they are much more rarely affected in their entirety, unlike wired systems. One of the main barriers, researchers and industry need to tackle in order for WSAN to become pervasive in this application domain is the lack of Quality of Service (QoS) support (reliability, dependability, security, etc), mainly due to their wireless nature. Finally, WSANs need to address the integration and representation problem that deals with how to make the information sensed by the network easily accessible and to provide users with a way of representing and querying the information collected by the WSAN.

All these challenges can be classified into three categories: high level programming abstractions, QoS and the integration problem. Regarding the first issue, there are currently widely used standards for WSANs, mainly at the data link layer such as IEEE 802.15.4 or 6LoWPAN. However, there is still no agreement on what programming abstraction to use in upper layers. Despite the fact that a huge number of different high level abstractions have been proposed, most of the applications are still programmed at a low level of abstraction and tailored for each specific application. Regarding the second point, although currently there are protocols that allow information to be reliably transmitted to a neighbor, more elaborate, dependable routing protocols are needed so that the network dynamically adapts to changing environments and possible failures. Also, these protocols are expected to provide the QoS requirements needed by modern applications like the CIP applications. Finally, the integration problem needs to provide solutions that make use of machine-independent protocols and that collect and store the data sensed by WSAN. It also needs to provide ways of querying and representing the information collected from the WSAN in a wide variety of devices such as mobile phones, PC, laptops, modern TVs, etc.

1.2 Contribution of this Thesis

This thesis aims to investigate each of the aforementioned challenges that hinder WSANS from becoming widely used. We analyze the problems that need to be tackled in each of these challenges and propose some solutions. The final goal is to provide WSAN developers and users with a middleware, called PS-QUASAR(Publish/Subscribe QUAlity of Service

Aware middlewaRe), that they can use to program these devices in an easy and non error-prone manner and allow QoS requirements to be handled automatically. In addition, a way of achieving communication between different devices, WSANs and a Supervisory Control and Data Acquisition (SCADA) is presented.

In general terms, this thesis contributes to mitigating the problems of actual WSAN technology by providing solutions at different levels. In particular these are:

- An extensive study of current MAC protocols, routing protocols and middlewares focusing on their application to the CIP problem. Desirable features are identified and the challenges are described and analyzed.

- A routing protocol for WSANs that support a many to many communication pattern, and that can handle QoS requirements. This protocol lays the foundation over which a publish/subscribe programming model is used. QoS requirements supported by the PS-QUASAR routing protocol are reliability, deadline and priority. Developers specify their QoS needs and the middleware automatically handles the requirements.

- A high-level programming abstraction based on the publish/subscribe paradigm that simplifies the development of programs for WSANs. Developers are not aware of the network topology or the way the routing protocols work. Publishers generate data about a specific topic. Subscribers declare their interest in certain topics. The middleware transparently delivers the information from publishers to subscribers which have declared their interest in the same topic.

- An analysis of the performance of common queue-based priority mechanisms by means of the PS-QUASAR middleware, as well as a study of the performance of the network under different workloads, in terms of reliability, delay and queue occupation using different network topologies and network sizes. This study aims to identify the importance of matching network level capabilities to data link layer capabilities and comments on the existence of a noticeable tradeoff between reliability and priority.

- A case study is presented to validate the PS-QUASAR middleware. The case study consists of a WSAN for monitoring a railway bridge. The application shows ways of coping with large amounts of data and deals with mobility issues. Specifically, passing trains are used as data-mules to collect the information sensed by the WSAN.

- A general architecture consisting of a SCADA that connects and collects information from multiple WSANs. Information is stored in a database and can be queried over the internet using a common browser. This architecture makes use of open source software and widely used technologies such as web services over HTTP.

1.3 Outline of this Thesis

This thesis is structured as follows:

Chapter 2: QoS in WSANs. Challenges and Open research issues. In this chapter, the concept of QoS for WSANs is defined and the different QoS parameters are described (Section 2.1). In addition, the most challenging aspects of WSANs are outlined (Section 2.2). Particular attention is paid to those challenges related to the QoS support and the CIP problem. Finally, the desirable properties of a CIP WSAN are analyzed in Section 2.3.

Chapter 3: Related work. This chapter introduces the related work. It has been divided into three different sections. Section 3.1 presents different programming abstractions that have been proposed to program WSANs. Section 3.2 studies the state of the art in routing, MAC protocols and middlewares that can handle QoS requirements and identifies a set of requirements WSAN protocols should meet in order for them to be used in demanding applications. The analysis focuses on the use of WSANs in CIP applications. Finally, Section 3.3 introduces the concept of SCADA, their use in combination with WSANs and existing solutions.

Chapter 4: PS-QUASAR middleware. In this chapter the PS-QUASAR middleware is presented. The middleware provides a publish/subscribe programming abstraction that simplifies the task of developing applications for WSANs. It also provides QoS in the communications between publishers and subscribers that can be specified using the API provided and a maintenance and routing protocol that automatically handles the communication tasks. The problem formulation and assumptions are presented in Section 4.1. The middleware is described in Section 4.2. Finally, the evaluation is carried out in Section 4.3.

Chapter 5: Matching data link and network communication layers. This chapter describes the need to match network level capabilities to data link layer capabilities and comments on the existence of a noticeable tradeoff between reliability and priority. The chapter analyzes and discusses the difficulty of providing a middleware supporting QoS, focusing on the influence of network traffic on network performance and the problem that common mechanisms to provide priority have in terms of delay reduction. The motivation of this analysis is presented in Section 5.1. The case study scenarios are explained and analyzed in Section 5.2. The evaluation is presented in Section 5.3.

Chapter 6: Case study: Railway Infrastructure Health Monitoring. In this chapter the PS-QUASAR middleware is examined by means of a case study that deals with challenges such as a large amount of sensed data that needs to be delivered, and mobility. The case study consists of a WSAN that monitors a CIP, more specifically a railway bridge. Passing trains are used as mobile data mules to collect the information sent by the sensor nodes. Section 6.1 describes the case study's motivation. The related work concerning this particular application is presented in Section 6.2. For the sake of clarity the related work of this case study has been included in this chapter and not in the general related work chapter, found in Chapter 3. The architecture and implementation details of the case study are presented in Section 6.3. The evaluation is carried out in Section 6.4.

Chapter 7: Integration of SCADA and WSANs. This chapter introduces an integration solution for SCADA/WSAN by means of a case study. The case study consists of a WSAN that monitors and controls an electrical power grid (Section 7.1). The architecture makes use of open source software and widely used technologies such as web services to seamlessly connect WSAN and the internet and make the information available to a wide variety of devices. The integration solution is detailed in Section 7.2.

Chapter 8: Conclusions and Future Work. In this chapter the main contributions and results of the thesis are presented. Conclusions and future work are presented in Sections 8.1 and 8.2, respectively. Some general comments about the thesis and this area of research are expressed in Section 8.3. Acknowledgements are presented in Section 8.4.

2

QoS in WSANs. Challenges and open research issues

The need for and importance of supporting QoS in WSANs have been identified in Chapter 1. QoS is a general concept which involves different parameters that need to be identified. In addition, several challenges and research issues need to be taken into account in order to provide protocols that support QoS. Section 2.1 studies the concept and importance of QoS in WSANs, emphasizing their applicability to the CIP problem . Section 2.2 presents the current challenges in the development of protocols that provide QoS for WSANs. Finally, desirable properties of a CIP WSAN are described in Section 2.3.

2.1 QoS Requirements in WSANs

QoS encompasses a lot of areas. Depending on the field in which it is used, the meaning can vary slightly. QoS can refer to the capability to provide assurance that the service

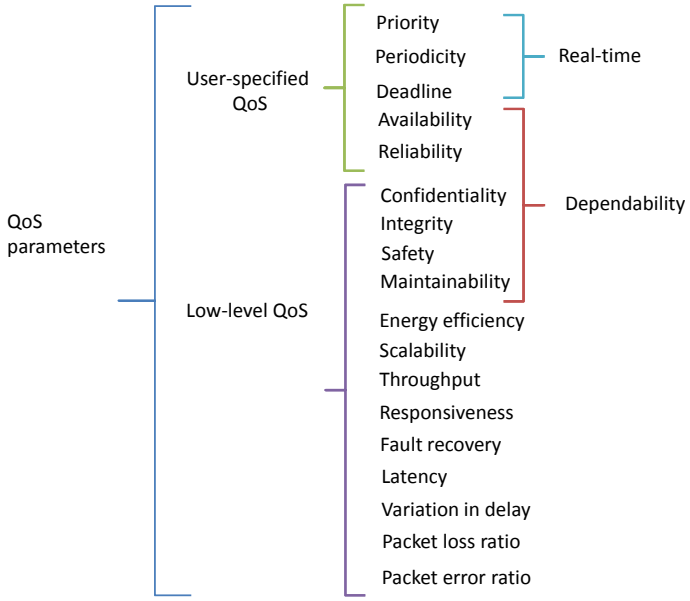


Figure 2.1: QoS parameter classification

requirements of applications can be satisfied [23]. QoS can also be defined as the network's ability to customize the treatment of specific classes of data. Finally, QoS can be seen as the resource reservation control mechanisms that can guarantee a certain level of performance of a data flow. This definition, contrary to the previous ones, refers to the mechanisms to achieve a certain service quality rather than on the service quality itself.

An application's QoS requirements are conveyed in terms of high-level parameters that specify what the user requires. These parameters can specify many different aspects of the system as the QoS term is very general. QoS parameters can also be used to measure the system performance and to control whether the QoS has actually been provided by it. We define the first set of QoS requirements as user-specified QoS and the latter as low-level QoS. By user-specified we mean the set of parameters that users can control to customize the communication between nodes in the WSAN. Low level parameters, on the other hand, measure features that are either provided by the middleware or somehow indicate its performance and cannot usually be modified by users. However, the boundaries between low-level and user-specified parameters are not well defined and other classifications are possible. For example, in existing proposals some of the low-level parameters are considered user-specified parameters and viceversa.

Figure 2.1 shows one of the possible classifications of different QoS parameters. Parameters in this classification are divided into user-specified QoS and low-level QoS parameters and special emphasis is put on parameters that measure QoS in the context of the CIP but a wider list of QoS parameters can be found in [23] [24]. In the end, the goal of a QoS parameter is to provide a way for users to specify certain needs regarding how the data in the network is handled, that is, the application requirements.

In CIP WSANs, dependability and real-time are very important parameters that need to be supported. The challenge is then, to design and develop protocols that provide them. So far, many attempts have been presented to provide some of these features but not enough effort has been dedicated to combine these protocols to provide all the parameters together in an effective way.

The parameters shown in Figure 2.1 are explained in the following points where their relationship with CIP is discussed.

- Priority: defines a way to assign different levels of importance to the data flows. This way, the more importance the data has, the sooner the system will try to process them. In WSAN CIP systems there are usually different levels of importance in the messages exchanged between nodes. For example, monitoring readings does not usually have the same importance as failure or attack notification events.

- Periodicity: refers to the quality of performing actions at regular intervals or periods. This behavior is common in monitoring applications which report periodic readings to the base station, as is the case for most of the CIP applications.

- Deadline: deadline is also known as maximum latency. It defines a maximum length of time in which some sort of task must be accomplished. Certain readings of the system may only make sense if received within a specific time span.

- Availability: the availability measures the degree to which a system is in a functioning condition. Critical infrastructures usually have high availability requirements because they must ensure the security of the system being monitored.

- Reliability: specifies the ability of the network to ensure reliable data transmission between nodes. Information in CIP needs to be reliably transmitted to make sure that

the possible error or important notifications generated in the system reach the desired destination.

- Confidentiality: as defined in ISO-17799, confidentiality involves ensuring that information is accessible only to those authorized to have access. This feature is essential in CIP systems which need to guarantee that information from the CI and also the monitoring information gathered by the WSAN are only accessed by authorized parties.

- Integrity: refers to the absence of improper system alteration. It is obvious that integrity is expected in CIP. In fact, maintaining the integrity of a critical infrastructure is one of the main functions of the CIP system.

- Safety: safety indicates the probability with which conditions that can lead to mishaps do not occur.

- Maintainability: maintainability specifies the ability to undergo modifications and repairs. WSANs are dynamic systems and the network topology and node number are expected to vary as time goes by. In CIP applications the system may need to be updated or repaired as a result of an attack or node failure or simply to accommodate new system requirements.

- Energy efficiency: deals with the correct use of the limited energy WSANs have. The configuration of the network should be optimal in that only the minimum energy should be consumed when monitoring and reacting in the CIP system, thus, improving the lifetime of the system, its effectiveness and justifying the cost of its deployment.

- Scalability: indicates the ability of the network to be enlarged and to handle growing amounts of work efficiently. WSANs are systems which need to be scalable because one of their strong points is precisely the use of a large number of nodes in the same network. If the scope of the monitoring network needs to be extended or new nodes are going to be deployed, good scalability will help the system perform much better.

- Bandwidth or Throughput: measures the amount of data flow transported within a certain period of time. Traditional monitoring applications do not generally generate

a large amount of data traffic. However, CIP requirements are a little bit more stringent in that sense, due to the high availability they require and to the critical nature of the system being monitored. Thus, WSAN platforms used in CIP should support enough throughput to cope with peaks in resource demands such as those displayed when an attack in the system occurs.

- Responsiveness: measures the ability of the network to adapt to changes in topology. Responsiveness will negatively affect reliability (as more control packets are needed to support it). Despite this, high responsiveness is a desired feature in CIP systems as node failures and attacks should have no impact on the general network behavior.

- Fault recovery: defines the ability of the network to recover from a failure in one or more of its nodes. This requirement is essential to guarantee an effective performance of the CIP control system which is in fact especially designed to cope with failures and attacks.

- Latency: measures the delay experienced in the system when performing a certain action, for example, packet delivery latency. Latency requirements vary depending on the application requirements and should be minimized as much as possible.

- Variation in delay: this term is sometimes, incorrectly, referred to as jitter. It is defined as the difference in delay between packets in a flow. Information received by the base station about the readings of the WSAN CIP system should be synchronized. For example, readings from an event that has produced an increase in temperature should be reported approximately at the same time by nodes close to the event.

- Packet loss and error ratio: these two parameters are often used when measuring WSAN QoS since WSAN communications are intrinsically error-prone.

Developing applications for WSANs is not an easy task and many different challenges need to be faced. Many of the them are common to all distributed systems but there are many others that arise from the intrinsic nature that these devices currently present. This chapter summarizes the most challenging aspects of WSANs focusing in those related to QoS and the CIP problem.

2.2 Challenges

2.2.1 Resource-limited platform

Current WSAN platforms have very limited resources because of the embedded character of the devices, their small size, the dynamicity they present and also because of their autonomous nature. Sophisticated protocols are needed to implement complex QoS functionality in this kind of environment.

One of the most restraining features of WSANs is the limited power source they have. The capacity of current batteries used to power sensor nodes is expected to grow over time, but not fast enough to satisfy the sensor node demands. All protocols developed for these kinds of platforms need to be designed to be energy efficient. Because the most energy-expensive operation involves communicating sensor nodes using the radio transceiver particular emphasis should be put on minimizing data communication.

If a node goes down as a result of a depleted battery, it is not always feasible to manually replace it. Alternatively, some energy harvesting techniques can be used in order to ensure an unlimited (under energy efficient protocols) energy source. Monitoring applications in WSANs usually relay all the information they get from the sensors to some sink nodes. As a result, memory consumption in the nodes of the networks will be unbalanced, which means, nodes near the sink nodes will receive more packet traffic than the rest. Some techniques need to be developed to balance the network traffic to the maximum extent possible. Some methods for obtaining energy efficiency are data aggregation, data fusion, duty cycling and time synchronization.

Other constraint resources to bear in mind involve the bandwidth, memory, processing capabilities and transmission power of the platform.

2.2.2 Data redundancy

One of the main features of WSANs is the high density with which nodes can be deployed. If proper algorithms are used, this redundancy can help the network to be more reliable and fault tolerant as failures in individual nodes do not affect the overall network behavior. However, redundancy affects the energy consumption of the WSAN as the same readings are gathered by physically close nodes. In this sense, data aggregation techniques can be

used to reduce the information transmitted over the network and to report related readings as only one.

In CIP, the redundancy is one of the most interesting factors that make WSANs useful for that goal. By deploying a high number of nodes in the environment, not only is it easier to get fault tolerance but also to balance traffic and energy consumption throughout the network for example by using duty cycling.

2.2.3 WSAN dynamism

WSANs are highly dynamic distributed systems. The state of the network is expected to change as nodes fail, are added or their internal state changes (from or to the sleep mode for example). Also, the connectivity graph can change as the communication range varies because of a change in the transmission power or due to external factors.

In addition, if node mobility is allowed, the protocols are much more complex. Designing protocols that allow node mobility and effectively provide QoS requirements is very challenging. The fact that the position of a node can vary over time makes it hard to route packets under strict QoS requirements. Routing tables are much more dynamic and neighbor discovery techniques become much more complex especially if the node mobility range is not bounded. CIP applications using WSANs, however, are usually not expected to be mobile.

2.2.4 Scalability

Many QoS-aware protocols have been presented in the scientific literature but most of them have only been simulated. Simulation is important because it allows general information to be extracted about the behavior of the algorithm and about its suitability in a real platform. However, real platform tests should be used whenever possible since performance results significantly vary among them. Furthermore, based on our own experience, to obtain good scalability is much more difficult if the tests are done on real platforms than in a simulation. This is primarily due to the assumptions simulators make in the communications between nodes. Real nodes show a much more unpredictable behavior that needs to be taken into account.

Approaches such as clustering can significantly help to improve the scalability of the

system. In such approaches the network is divided into isolated clusters which can not communicate with each other. This way, traffic in the network is considerably reduced and new cluster additions are independent from existing ones.

Despite all efforts made towards improving scalability, current WSAN size is not greater than around a thousand nodes [25]. Further research will make it possible to see networks with tens of thousands of nodes, but as envisioned by the scientific community this number is expected to be much higher in the future.

CIP WSANs are thought to be scalable as the number of nodes is expected to be high. This high density is used to guarantee fault-tolerance and to provide QoS.

2.2.5 N-to-N communications

Communications in WSANs are not restricted to sensor reading delivery between sensors and sink nodes. More complex communications are also used in many proposals which involve an N-to-N communication pattern between all devices in the network. This is particularly true in networks which not only report information to the sink nodes but also can be managed and accessed at will by users. Although N-to-N communications improve the system flexibility and capabilities, this kind of communication makes it more difficult to guarantee QoS as it can be carried out between any pair of nodes at any time. Protocols need to bear that in mind and try as far as possible to balance the traffic and communication between nodes.

In CIP applications N-to-N communications can be used for example to exchange messages between a set of sensors and actors to reach an agreement on how to keep track of an abnormal event.

2.2.6 Multiple types of traffic

It is usual for some applications to share the same WSAN, each of them having their own requirements. Thus, the platform needs to support different kinds of traffic and should adapt itself to the needs of the applications, which may be at some point in conflict with each other. In the case the application requirements can not be met, the application should be notified.

2.2.7 QoS models

Although the main subject of research in this context is to actually devise and implement solutions that provide QoS, the study of methods that simplify the task of specifying these requirements is also important. Current solutions are based on parameters that the user needs to specify and that influence the QoS configuration of the platform (e.g. priority and deadline). The configurable QoS parameters offered by the platform should be simple and the unit used for every parameter should follow a standard, if possible. Also the number of parameters offered should constitute a small but complete set that allows the developer to specify all possible requirements in the platform. Lastly, it is important to bear in mind that the programming abstraction used for the system is going to influence the way in which QoS requirements are specified.

2.2.8 Real-time and reliable platforms

The distributed nature of this technology makes it impossible to provide 100% reliability in the delivery of messages. However, by means of protocols, the delivery probability can be raised to acceptable levels. To provide such features, complex protocols need to be used, which in turn have a negative influence on other aspects of the system such as energy efficiency or scalability. Most of the current solutions only provide soft real-time assurance. This is due to the limited real-time and reliable capabilities of the underlaying platform over which the application runs. In order for a protocol to provide hard real-time properties, the operating system of the platform where it runs needs to support it. An example of a platform of this kind is uC/OS-II [26].

Although, hard real-time is usually the appropriate choice for CIP, sometimes using a soft real-time platform with good QoS support algorithms may suffice.

2.2.9 Energy Efficiency vs QoS

It is obvious that the more tasks need to be done, the more time they will consume. In this sense, providing QoS requires additional effort in terms of computation, memory and communication. All these things have an important effect on the energy consumption. It would be interesting to study under what circumstances this extra energy consumption is worth it and moreover to adapt the network to dynamically choose the best configuration

to provide the QoS requirements specified. More generically, the study of trade-offs is important in the way that it allows the network to adapt itself to the application needs ensuring that no extra computation or communication is wasted.

This study is particularly interesting in the CIP problem where the availability of these kinds of systems needs to be high and the life time of the network needs to be maximized.

2.2.10 Standardization

As the WSAN field is still being developed, some standards have appeared which try to establish common ways of providing functionality at physical, link and even network layer (ZibBee [27], 6LoWPAN [28], WirelessHART [29]). However, standard QoS handling procedures are not well defined yet. This is due to the relatively little research on the area of QoS in WSANs. One of the challenges to overcome is to achieve a generic way of providing QoS that satisfies the needs of all users.

Currently, there is much interest in the development of IPv6 support over WSANs in an attempt to standardize the protocols in WSANs, to reuse the knowledge about IP networks and to connect WSANs to existing wired and wireless networks throughout the world. The 6LoWPAN protocol, for example, defines the encapsulation and header compression mechanisms that allow IPv6 packets to be sent and received over IEEE 802.15.4 based networks

The use of IPv6 in CIP WSAN is interesting. This would allow the network to be queried over the internet, for example from a SCADA system that can be used to monitor the state of the CI and the WSAN. The choice of a proper standard greatly simplifies the task of achieving interoperability between this SCADA system and the WSAN.

2.2.11 Cross layer architecture

The layered design of communication protocols such as TCP/IP has given really good results in terms of simplicity, scalability. The layered design makes it possible to change or update parts of it without affecting the whole protocol. However, the inflexibility and suboptimality of this paradigm result in poor performance for multihop wireless networks in general and more specifically for WSANs, especially when the application has high bandwidth needs and/or stringent delay constraints [30]. Cross-layer design allows communication between layers (such as physical and network layers) which traditionally have

been independent in order to exchange more information that can be used to improve efficiency. The traditional layered approach should be used if requirements can be met without applying a cross-layer design. However, WSAN platforms usually resort to cross-layer techniques in order to maximize the performance.

2.2.12 QoS protocol integration

Many protocols have been proposed that deal with some QoS aspects. Many of these works describe a protocol that has been evaluated by means of a simulator. In order to actually confirm that all this work can be applied to real examples we think that it is useful to make a greater effort to try to integrate different algorithms and strategies in a complete system and test its performance using real devices.

Although simulators have their place, we believe that what really validates the usefulness of a set of proposals comes as a result of developing a complete system that takes into account the knowledge gathered in the field. A protocol that seems to perform well may not work well when used in combination with others and using real devices.

2.2.13 QoS monitoring and application development

Considerable effort is being invested in developing ways of providing QoS. However, it is also really important to be able to monitor and study the results taken from the WSAN. Techniques need to be designed that allow the network state to be queried and to keep track of the way it evolves. Also, development tools need to be refined. In this sense, node debugging and node code deployment are still arduous tasks and further development is expected to be done in the area of WSAN simulators. Other areas such as the implementation of IDEs that allow programmers to simplify the tasks of developing WSAN applications are much more developed.

2.3 Desirable properties of a CIP WSAN

Several MAC, routing, hybrid protocols and middleware are analyzed in Chapter 3 and some ideas taken from them. In order to better illustrate the main features expected in a WSAN application for the CIP, Figure 2.2 presents a possible classification of the main

desirable features at each communication layer. Not all QoS parameters presented in Figure 2.1 are included in Figure 2.2 only the main features a CIP system needs. Besides, some of the parameters can be offered at different layers as in the case of reliability but it has only been included in the layer that usually provides it. This classification can vary depending on the specific needs of each scenario but it is expected to be useful as a guideline when implementing a WSAN for a CIP system.

The physical layer (first layer in the OSI model) is platform dependent and therefore has not been included in the figure.

Layer 2 accommodates the MAC layer. This layer has a great impact on the WSAN's performance and therefore needs to be energy efficient. Also, all upper layers rely on the functionality provided by the MAC protocol. This means that ideally the MAC layer should handle different traffic classes (manage priority). This property, however, can also be provided at network layer as many proposals in the literature do. The MAC protocol cannot be dependent on the underlying distribution of the nodes, that is, it should be flexible enough to efficiently handle different network topologies. Finally, the use of cross-layer interactions between the MAC protocol and the network protocol seems an interesting approach to improve performance.

The routing protocol stands at the network layer and allows multi-hop delivery of information between motes. Because important nodes in centralized algorithms can be subject to attack, it is advisable to opt for a non-centralized one. Fault tolerance and reliability are two important parameters that the routing protocol needs to provide in CIP systems. In some approaches these two parameters are offered in the MAC layer or even at transport layer rather than in the routing protocol. Critical data needs to be delivered on time in order to act against possible failures or attacks, so bounded delay is also an important task of the network layer.

Layers above the network layer provide a way for developers to specify their needs and the behavior of the application. On the one hand, a higher level of abstraction is needed, for example, in the form of a middleware. This allows programmers to simplify the task of programming application and leaves certain repetitive tasks, such as communication or node discovery, in the hands of the middleware, making programs less error-prone. On the other hand, QoS requirements should be specified in a flexible way that allows programmers to map their QoS needs to an appropriate network configuration.

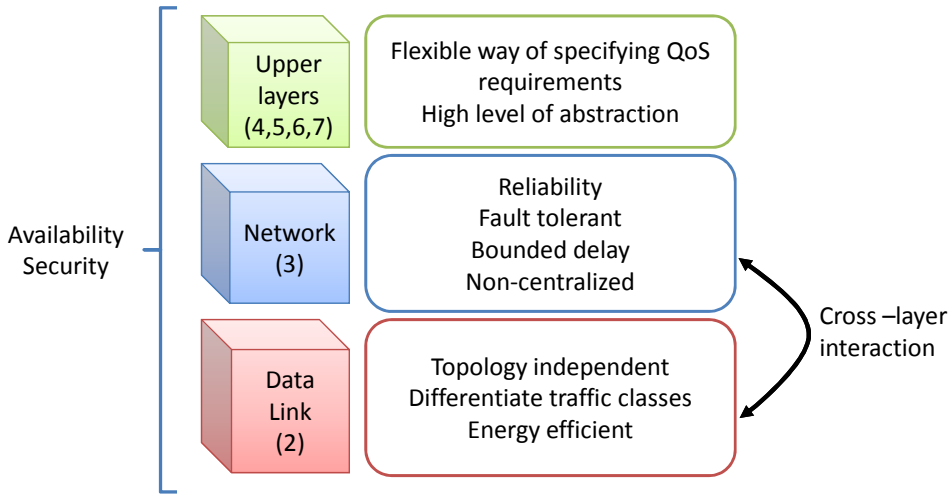


Figure 2.2: CIP WSAN desirable features

Finally, certain properties that need to be provided are not localized in any concrete layer. Among them the most important are availability, which ensures that the network is constantly monitoring the CIP system, and security that protects the system against undesired access, failures or attacks.

3

Related work

This chapter analyzes related work in three different domains. Section 3.1 surveys different programming abstractions and middlewares that have been proposed to rise the level of abstraction with which WSAN applications are programmed. Section 3.2 extensively surveys existing MAC protocols and routing protocols, focusing on their capacity to cope with QoS in the communications between nodes. Finally, Section 3.3 briefly discusses the integration problem.

3.1 Programming abstractions and middlewares

Programming abstractions and middlewares are used to simplify the task of developing applications. They hide low level concerns such as packet management, queue management, QoS handling and allow developers to focus on specifying the program behavior rather than having to implement the same low level tasks every time. Many different approaches have been proposed to raise the level of abstraction used to program sensor nodes. This section

Name	Energy efficiency	Scalability	QoS parameters				Features
			D	R	PR	PE	
Agilla	N/A	N/A					Mobile Agent-based system
Mate	Moderate	N/A		X			Virtual machine-based system
SwissQM	N/A	Good				X	Stack-based integer virtual machine
TinyDB	High	N/A			X	X	Acquisitional query processing system
SINA	N/A	N/A					Acquisitional query processing system
Kairos	High	N/A			X	X	Use of a macroprogramming language to specify the application behavior
Regiment	N/A	N/A			X		Use of a macroprogramming language to specify the application behavior
TeenyLime	High	Good		X			Tuple space abstraction, split-phase operations
Oasis	N/A	Good					Multilayer development process, service-oriented architecture, object-centric programming
TinySOA	High	Good				X	Service-oriented middleware, semantic-aware routing
USEME	Moderate	Good	X	X	X	X	Combination of macroprogramming and node-centric programming, service-oriented architecture
ATaG [31]	N/A	N/A				X	Mixed imperative-declarative approach, data driven program flow
Tenet [32]	High	Good		X		X	Tiered network, task dissemination, reliable transport, rate adaption
Milan [33]	High	N/A					User-specified Qos requirements that Milan automatically manages

Table 3.1: WSAN QoS middleware summary (D: deadline; R: reliability; PR: priority; PE: periodicity)

covers the main categories. The summary of the main features of the middlewares covered in this section and some other existing proposals is depicted in Table 3.1, together with the QoS parameters provided by each proposal.

3.1.1 Mobile agents

An agent-based model makes migration decisions autonomously. The key to this approach is to make the application as modular as possible to facilitate its injection and distribution through the network. One of the most important ideas that follows this approach is Agilla [34]. The idea behind Agilla is to initially deploy a network without the need for a previously installed application. Agents that implement the application behavior can later be injected, effectively reprogramming the network. Because each agent is executed autonomously and multiple agents can simultaneously run on a node, multiple applications can co-exist. Agilla presents a compromise solution between flexibility and power consumption due to the transmission of agents by the network. Agilla uses acknowledgments and timers to retransmit if packets are lost but does not allow user specification of QoS constraints.

3.1.2 Virtual machines

Proposals such as Mate [35] are based on a virtual machine that runs on top of the devices. The use of a virtual machine allows the virtualization of real hardware, bytecode interpretation and intermediate program representation. Mate provides a bytecode interpreter running on top of TinyOS [36]. The idea behind this virtual machine is that applications are usually composed of the same set of services and sub-systems, combined in different ways. By offering these services, program size can be reduced considerably. For this reason the virtual machine offers a predefined set of instructions that are used to program the applications.

Another virtual machine based middleware, called SwissQM is presented in [37]. SwissQM has a small footprint and specializes in data acquisition and data processing. It follows a multi-source to single-sink communication pattern that is materialized by means of spanning-tree whose root is the sink. SwissQM provides a stack-based, integer virtual machine that runs over TinyOS. SwissQM also provides functionality that permits data aggregation, program dissemination and topology management. Program fragments in the dissemination protocol are handled in a reliable way by means of timeouts and message snooping.

3.1.3 Database

Examples of this category are TinyDB [38] and SINA [39]. TinyDB is a query-processing system that extracts information from the data collected by the WSN using the underlying operating system, TinyOS, and a controlled-flooding approach to disseminate the queries throughout the network. SINA is a more complex database approach than TinyDB since it not only permits SQL-like language for expressing queries, but also provides other functions which are outside the scope of traditional database systems. SINA incorporates two robust mechanisms: hierarchical clustering allowing scalability and an attribute - based naming scheme based on an associative broadcast to manage the spreadsheets.

3.1.4 Macro programming

Traditional applications are composed of different programs each of which is located in a node of the network. Applications are therefore programmed by specifying the local behavior of all nodes in our network. Proposals such as those by Kairos [40] and Regiment [41]

have a different approach. Applications can be created by specifying the global behavior of the network. This behavior will be then translated into different pieces of code which will be used in the nodes. The programming tasks are therefore at a higher level of abstraction.

In Kairos the global behavior of the network is specified by means of a set of language independent programming primitives. With these primitives we can read and write variables in nodes, iterate through the one-hop neighbors of a node and address remote nodes. With the program created using these primitives and using Kairo's compile-time and runtime systems we can obtain a node-specialized version of the program behavior that will be executed in all nodes. Regiment is a macroprogramming language that allows programming a collection of nodes as a single entity. In Regiment, a sensor network state is represented as time-varying *signals*. Signals might represent sensor readings in an individual node, the state of a node's local computation, or aggregate values computed from multiple source signals. Regiment also supports the notion of *regions*, which are spatially distributed signals.

3.1.5 Tuple space

The coordination needs in WSNs and WSANs have attracted the attention of the Coordination paradigm community. More specifically, different coordination models and middleware based on the Linda abstract model [42] have appeared in the area of sensor networks. Linda can be considered the most representative coordination language. It is based on a shared memory model where data is represented by elementary data structures called tuples, and the memory is a multiset of tuples called a tuple space. Examples of this type of middleware are TinyLime [43] and TeenyLime [44]. In TinyLime, sensors are sparsely distributed in an environment, not necessarily able to communicate with each other, and a set of mobile base stations (laptops) move through the space accessing the data of nearby sensors (single-hop). Each base station owns a tuple space and federated tuple spaces can be established in order to communicate and synchronize several base stations and some client hosts. TeenyLime is an evolution of TinyLime to be applied in WSANs. As in TinyLime, the core abstraction is the transiently shared tuple space, but in this case the spaces are physically located in the sensors themselves.

3.1.6 Services

In this category functionality in the network is accessible as a service and it is in this way that communication is achieved between nodes. The programmer can specify the execution flow by compacting these services together. The use of services provides a high level of abstraction since programmers specify behavior by creating a program that makes use of different services but the programmers do not have to worry about the communication or protocol tasks used for this goal.

Oasis [45] proposes a programming framework which provides abstraction for object-centric, ambient-aware, service-oriented sensor network applications. Object-centric programming is achieved by providing the application developer with a means for specifying the global behavior of the network using finite state machines. TinySOA [46] presents a service-oriented middleware. The main entity in TinySOA is the service which is considered as a computational component that has a unique identifier and it is invoked asynchronously. Users can access the information via services by querying base stations or directly querying individual nodes in the network. Queries are divided into task and event queries and have the form (Event-Condition-Action-Spatial scope-Temporal-Scope). USEME [47] is a service-oriented middleware framework for WSANs. In USEME each service is a composition of a number of ports, each of which is a bi-directional interface comprising synchronous and asynchronous communication channels. The application general behavior is specified using the USEME abstract language and is later automatically translated into a set of templates. These templates need to be filled with platform-dependent code to specify the behavior of each operation defined in the previous step. Real-time communication constraints can be associated with each command or event helping to ensure timely network operation.

3.1.7 Publish/subscribe

The publish/subscribe model is a widely known communication paradigm to asynchronously communicate nodes in a distributed system. Two main entities exist in this approach: 1) publishers that produce data of a certain topic 2) subscribers that consume data of a specific topic. The goal of the publish/subscribe system is to transparently send the data from publishers to subscribers that have declared their interest in the same topic. In this paradigm

Name	Multiple subscribers	Support Actors	Scheme	QoS	Fully distributed
Mires	No	No	Topic-based	No	Yes
Wireless sensor networks based on publish/subscribe messaging paradigms	No	No	Unknown	No	Yes
Fault-tolerant wsn routing protocol for the supervision of context-aware physical environments	No	No	Unknown	Reliability	Yes
MQTT-S	No (only outside the wsan)	Yes	Content-based	Priority, deadline, reliability	Yes
Quad-PubSub	Yes	No	Unknown	No	Unknown
TinyMQ	Yes	No	Content-based	No	Yes
A Publish-Subscribe Middleware for Real-Time Wireless Sensor Networks	Yes	No	Unknown	Priority	Yes
TinyDDS	Yes	No	Content-based	Priority, deadline, reliability	Yes
Clustered Publish/Subscribe in WSANs [48]	Yes	Yes	Unknown	No	Yes
A policy-based publish/subscribe middleware [49]	Yes	Yes	Topic-based	Reliability	Yes
PS-QUASAR	Yes	Yes	Topic-based	Priority, deadline, reliability	Yes

Table 3.2: Publish/subscribe proposals summary

publishers and subscribers do not know each other. The system is in charge of automatically delivering data produced by publishers to the corresponding subscribers.

As our proposal is based on a publish/subscribe model, this section makes a more extensive study of existing proposals than the previous ones focusing on the capability of protocols to provide QoS.

A generic overview of publish/subscribe protocols is presented in [50] where the different publish/subscribe variations are classified and discussed. [51] deals with the use of the publish/subscribe in sensor networks. The existing challenges and possible techniques to overcome them are presented. In [52] the different techniques that can be used to implement a publish/subscribe protocol in sensor networks are studied together with the possible primitives in these kinds of systems. Single-subscriber and multiple-subscriber protocols are covered in Sections 3.1.7.1 and 3.1.7.2, respectively. A summary of all presented protocols and middlewares, with the one presented in this thesis (PS-QUASAR) is shown in Table 3.2. The main features of each of the proposals covered in the related work are summarized in this table.

3.1.7.1 Single-subscriber routing protocols

Mires [53] is a middleware for WSANs that uses a publish/subscribe paradigm to communicate the sensor nodes with a sink node. Communication between publishers and subscribers takes the following steps. First, all nodes in the network must send the information to advertise their available topics. Once this information is received the sink node selects the topics it wants to subscribe to by sending a message to the corresponding nodes. Finally, sensor nodes receiving this message can start sending information about the selected topics. The multihop routing protocol used to exchange messages is not covered in the paper. Also, Mires does not directly support QoS. In PS-QUASAR the implementation of the paradigm is substantially different. All nodes in the network have access to the existing subscribers and can send information to them without a two-step protocol.

A simple publish/subscribe system is presented in Wireless sensor networks based on publish/subscribe messaging paradigms [54]. The protocol only allows many-to-one communication which means that there is only one subscriber (the sink node). The protocol is initiated by the sink node which broadcasts the topics it is interested in. Nodes receiving the packet will publish information back to the sink. This means that communication is always initiated by the sink node and requires two communication phases (one in each direction) to work. This protocol does not handle QoS and does not cope with fault tolerance.

In [55] two routing protocols based on the publish/subscribe paradigm are presented. The first one uses the Bellman-Ford algorithm to locate the sink node which is the only subscriber. Once a routing tree is formed subscriptions are sent using flooding. Information is sent to the sink using the shortest paths to the root of the tree. A recovery protocol that selects a new path to the sink if a failure in a node in the current path is selected is also presented. This protocol relies on ACKs to detect the failures. Finally, a clustering mechanism is proposed where nodes with the highest remaining energy aggregate information from the neighbors and relay it to the sink. This protocol always uses ACKs to send information from node to sink which can be energy consuming.

MQTT-S [56] adapts the MQTT protocol (based on the publish/subscribe paradigm) to sensor networks. Sensor nodes are publishers whereas the external applications receiving the data are subscribers. The protocols need the assistance of an external broker which controls how to match the data generated by the network with the corresponding applications.

MQTT defines different levels of QoS that provide different reliability schemes and also supports the use of actors. MQTT does not specify any routing techniques; it assumes a TCP/IP like protocol. The main different to PS-QUASAR is the use of an external broker that controls publisher/subscriber matching.

3.1.7.2 Multiple-subscriber routing protocols

A publish/subscribe protocol for location-aware WSANs called Quad-PubSub is presented in [57]. The protocol assumes that all nodes are aware of their position and that of their neighbors. The network area is divided into geographical scopes controlled by special nodes called "Event brokers" (EBs). Each event broker offers access to information generated by publishers in its area. After that, subscribers are linked to those EBs that can serve them with their set of interests. The paper does not however clarify how to implement the different steps of the protocol in a real device. This protocol does not handle QoS.

TinyMQ [58] presents a content-based publish/subscribe middleware for wireless sensor networks. TinyMQ uses an overlay network together with a naming scheme based on binary strings to connect publishers with subscribers. First the network is partitioned into several tree-based clusters and each node is given a binary address by means of a maintenance protocol. This binary address is such that it allows routing to be made based on the addresses and distances to root nodes. TinyMQ does not handle QoS.

A publish/subscribe middleware for real-time wireless sensor networks is presented in [59]. The protocol makes use of broadcast communications to notify of the different interests and also to deliver the information. QoS support is achieved by using two packet queues: one for information requiring QoS and the other one for normal packets. Packets are processed according to their priority value. The protocol has been tested in a 40-node network with 2 subscribers and 4 publishers. The paper neither clarifies how paths to relay the information are obtained nor how failures are dealt with. PS-QUASAR uses a similar approach to cope with QoS requirements.

DDS [60] or Data distribution service for real-time systems is a specification developed by the OMG consortium of a middleware for distributed systems based on the publish/subscribe paradigm. DDS allows users to specify Quality of Service (QoS) parameters that will apply to the communications between nodes such as reliability, deadline, durability or transport priority. QoS expected/offered at publisher/subscriber side respectively needs

to be compatible in order for the communication to take place.

TinyDDS [61] presents an implementation of the DDS standard for WSANs located at transport layer that can work on top of different network layer implementations. TinyDDS, does not focus on a specific routing protocol to be used but rather offers a layer design pattern that allows flexible use of different protocols. TinyDDS offers a complete solution to communicate not only nodes in the same network but also different TinyDDS networks by means of a set of gateways. Although the DDS standard is a rather simple standard, we believe that some parts can be optimized to cope with the WSAN limitations. PS-QUASAR offers a simpler programming model.

3.2 QoS in MAC and routing protocols

In the introduction, we mentioned that as applications become more complex, the expected demands on the platform supporting them also increase. If the platform itself does not directly meet the requirements of the application, protocols and tools need to be used to manage the hardware in order to provide the required functionality . Intense research has been carried out in areas such as architecture, protocol design, energy conservation and locationing but QoS in WSANs is still a largely unexplored field of research [62]. In applications such as nuclear reactor control, battlefield surveillance or more generically in critical infrastructure protection (CIP), it is essential to guarantee certain levels of QoS.

Critical infrastructures as defined in the glossary of [63] are “Infrastructures which are so vital that their incapacitation or destruction would have a debilitating impact on defense or economic security.” The task of ensuring their availability and correct functioning under different scenarios including failures and attacks is therefore of utmost importance. Examples of critical systems are water supply, emergency systems, government services, electrical power or telecommunications. Proof of the growing interest in this field is that the number of initiatives dealing with CIP such as the European project WSAN4CIP [64] has considerably grown in the last five years. There are two main reasons for the increasing interest in the CIP. First the new risks which have arisen as a result of the technological revolution and secondly, the 9/11 terrorist attacks. Both of these expose the fragility of modern societies caused by the interdependence of information networks [65] and highlight the importance of searching for new ways to improve and guarantee the resilience of this vital

systems.

In this respect, WSANs have the potential to become an integral part of the protection of CIPs. Their distributed nature makes them particularly suitable against failures and attacks as they are much more rarely affected in their entirety, unlike wired systems. The use of WSANs in this context can easily extend the existing sensing capabilities of the system in a cost-efficient manner as well as providing important features such as security. However, the main attraction of this technology for the CIP is the autonomous character of each node. Failures and attacks in individual nodes do not affect the general behavior of the network. Since the density of nodes deployed to monitor critical applications is expected to be high, this redundancy can be used to provide a high fault tolerance and reliability when monitoring a scenario.

The possibilities of WSANs have been acknowledged as promising for the CIP field. Proof of this is that the U.S. Department for Homeland Security stated, in the 2004 National Plan for Research and Development in Support for CIP, that one of the strategic goals was “to provide a National Common Operating Picture (COP)” for Critical Infrastructures, where the core of the systems would be an intelligent, self-monitoring, and self-healing sensor network. The Australian government, by means of the Cooperative Research Center for Security (CRC-SAFE), is examining and developing solutions to security problems in CIP systems, including WSANs [66].

However, in order to fully integrate WSANs as an integral part of CIP, several open issues and challenges need to be adressed such as the development of abstraction models to program WSANs and protocols that support QoS. Unfortunately researchers have not reached an agreement on how to abstract the programming tasks for these devices and, much less to provide QoS support.

This section, therefore, aims at providing an analysis of the state of the art on QoS in MAC and routing protocols.

3.2.1 MAC layer

The MAC layer is a sublayer of the second layer in the OSI model. It provides addressing and channel access control mechanisms which allow several nodes in the WSAN to communicate within a multi-point network. Traditional MAC protocols do not consider energy-efficiency as a crucial design point. For this reason, new protocols and adaptations

of existing ones have been presented and designed specifically for WSANs. They try to overcome the WSAN restrictions by tailoring the MAC protocol properties to the special needs of WSANs. The principal QoS parameters provided at the MAC layer are throughput, average packet delay and transmission reliability [67] but they are neither configurable nor accessible to the application developer. For further details about existing MAC protocols several surveys have already been presented [68][69][70].

S-MAC is presented in [71] and is one of the most studied MAC protocols for WSANs. This MAC protocol introduces the use of a low-duty-cycle scheme, adaptative listening, message passing and other techniques to improve its energy efficiency with respect to, for example, IEEE 802.11. Low-duty cycle is achieved by coordinating sleep periods between nodes which greatly improves energy efficiency but affects the latency. In order to avoid this as much as possible, adaptative listening is used. It consists of letting nodes overhear their neighbor's transmissions for a short period of time at the end of the transmission instead of being asleep the whole time. Finally, long packets are fragmented into small fragments and transmitted in a burst. All these measures aim at improving energy efficiency but result in a subtly negative effect on the throughput.

T-MAC [72] improves some of the features of the S-MAC protocol to improve its energy efficiency (up to a factor of 5 in scenarios with variable load) at the expense of obtaining lower throughput. Its main feature is the use of an adaptive duty cycle which dynamically adjusts the length of the active time by means of a timeout. The active time is the period of time in which each node is awake and ends when no activation event has occurred for a predefined time.

B-MAC [73] improves the throughput and energy efficiency of S-MAC and T-MAC proposals by using low power listening and elaborated clear channel assessment techniques such as the use of outliers to detect channel occupancy.

DMAC [74] is a MAC protocol for WSANs optimized for data gathering trees (uni-directional communication from sensors to a sink). DMAC duty cycles are dynamically adjusted on the basis of the traffic load. By assuming that the communication is carried out from sensors to nodes, DMAC can optimize the message delivery along a branch of the tree by properly scheduling the sensors. This schedule avoids the presence of sleeping nodes in the path between a sensor and the sink node.

PEDAMACS [75] is a TDMA MAC protocol for multi-hop networks. One of its main

features is the use of a high-powered access point to synchronize the nodes. Besides, communication schedules are calculated by the access point using the network information and later transmitted to nodes. The results show that the energy efficiency and delay bound obtained outperform those of a traditional MAC protocol without sleep cycles but the use of a centralized access point with access to all nodes in the network may not be suitable for all applications.

In Z-MAC [76] an interesting hybrid approach is followed. Sensors are assigned a time slot but they can also utilize another sensor's slots by means of a CSMA mechanism. The choice of which mechanism to use is based on network traffic. In low contention scenarios, Z-MAC behaves like CSMA. However under heavy network traffic, it is similar to TDMA. Z-MAC is specially effective in terms of throughput under high contention scenarios outperforming B-MAC at the expense of a higher energy consumption.

IEEE 802.15.4 [77] [78] is a standard which specifies the physical layer and media access control for low-rate wireless personal area networks (LR-WPANs). It has been established as the most suitable, but it is still not an optimal standard for WSN applications [67]. SunSPOTs motes [79] for example implement this standard. Networks can be built as either peer-to-peer or star networks. However, every network needs at least one coordinator. IEEE 802.15.4 can operate in beacon enabled and in non-beacon mode. In non-beacon mode a CSMA/CA approach is used that is not collision-free but it is suitable for many applications. On the other hand, in beacon mode the protocol makes use of a superframe to transmit data. This superframe is composed of three parts: the beacon, a contention access period and a collision free period (CFP). It is in the CFP part where guaranteed time slots (GTSs) can be allocated in order to transmit data with time critical requirements at the cost of increasing energy consumption.

The QoS-based MAC protocol [80] follows a CSMA/CA approach which not only adapts to application-oriented QoS, but also attempts to conserve energy without violating QoS-constraints. This is achieved by adaptatively adjusting the contention window-size depending on both the traffic and wireless channel characteristics. In this respect, duty cycle is dynamically adjusted to minimize energy consumption while satisfying QoS requirements. The proposed protocol differentiates between different traffic classes in order to prioritize them depending on their importance.

A priority-based mac protocol called PQMAC is proposed in [81]. Priority is assigned

to data according to the requirements they have. High priority data is sent more quickly by increasing the listen time (time where the mote is in an awake state), using an advanced wake-up scheme and using a multi-queue packet transmission architecture. The wake-up scheme (normal or advanced) depends on the network traffic conditions.

The Q-MAC MAC protocol makes it possible to differentiate network services by using priorities. The priority is assigned to each packet on the basis of their content, the number of hops it has already traveled in the network and an energy metric. An inter-node scheduling scheme is carried out by means of a contention-based scheme with a RTC-CTS phase (ready to send/clear to send). Performance results show that energy efficiency is similar to that of S-MAC but providing flexible differentiation between service classes.

In [82] a protocol called MMAC that takes into account mobility in sensors is presented. MMAC is a scheduling-based protocol that schedules sensor nodes depending on traffic information and the mobility pattern they present. MMAC assumes that the sensor nodes are aware of their location. This information is used to predict the mobility pattern of nodes and to adjust MAC frame time, transmission and random-access slots accordingly. Results obtained show that MMAC performance is similar to that of other existing MAC protocols such us S-MAC when there is no mobility but outperforms them when mobility is present.

a dual-mode real-time MAC protocol is presented in [83]. This MAC protocol can work in two different modes. In protected mode, the network is structured in cells which provide a reliable collision-free communication mechanism. The cell structure is managed at initialization. On the other hand, in unprotected mode, cells are not used and thus the transmission speed toward the sink is near optimal although collisions are possible. At initialization, the network runs in unprotected mode but can switch to protected and back to unprotected depending on the number of collisions. This two-mode switching approach makes it possible to guarantee the worst case delay bound and also good average performance. Nodes are assumed to know their location, maximum communication range and radio interface bandwidth.

Table 3.3 shows a summary of all MAC proposals surveyed in this section. This table is based on the one that can be found in [84] and has been completed to take into account all the MAC protocols presented.

Most of the existing MAC proposals only provide best effort message delivery. Real-time MAC protocols such as I-EDF or PEDAMACS, on the other hand, make strong as-

Name	Type	RT type	Topology Dependent	Energy efficiency	Handle different traffic classes	Scalability
S-MAC, T-MAC, B-MAC	CSMA/CA	best effort	no	high	no	good
DMAC	slotted contention-based	best effort	tree structure	moderate	no	good
Z-MAC	Hybrid (CSMA, TDMA)	best effort	no	high	no	good
PEDAMACS	TDMA	HRT	no	high	no	low
MMAC	Hybrid (slotted contention-based, TDMA)	best effort	no	high	no	good
Q-MAC	CSMA/CA	best effort	no	high	yes	good
Dual-mode MAC	Hybrid (FDMA, TDMA)	HRT	cell structure	N/A	no	moderate
IEEE 802.15.4	slotted CSMA/CA, GTS	best-effort/HRT	no	moderate	yes	good
QoS-based MAC protocol	CSMA/CS	best-effort	no	high	yes	good
PQMAC	CSMA/CA	best-effort	no	high	yes	good

Table 3.3: WSAN MAC protocols summary

sumptions such as the use of a cell topology architecture or the need for a high-powered access point to synchronize nodes, respectively. These assumptions make CIP WSANs less secure as an attack or failure in these high-powered access points may cascade to a large part of the network. IEEE 802.15.4 provides a compromise solution, flexible enough to allow different node topologies and with the possibility of supporting real-time QoS constraints. However, these advantages are provided at the expense of showing a moderate energy efficiency. In that regard, trade-off analyses need to be carried as explained in Section 2.2.9, to study the balance between energy efficiency and QoS provisioning. In any case, IEEE 802.15.4 seems a reasonable MAC candidate for use in a CIP system. As a matter of fact there has been increased interest in ZigBee (that sits over the 802.15.4 standard) for building automation and industrial controls as stated in [27]. In addition, the choice of a proper MAC protocol for a CIP WSAN system is partially based on the chosen routing algorithm and how the combination of both protocols performs. Finally, in this context, it is desirable for the MAC protocol, to be able to differentiate between different traffic classes, that is, to give priority to data which is more important like PQMAC, QoS-based MAC protocol, Q-MAC or IEEE 802.15.4 do. This feature, however, on many occasions is offered at network layer. Additional constraints will also influence the choice of the MAC protocol. For example, MMAC is the only surveyed MAC protocol which explicitly deals with this issue.

3.2.2 Routing protocols

The network layer is the third layer in the OSI model. The network layer is responsible for end-to-end (source to destination) packet delivery, a task which involves routing information through sensor nodes to the destination. Most of the real scenarios require the WSAN

to support multi-hop routing, that is, to be able to route messages between two nodes which can not directly communicate with each other.

In order to successfully route messages several issues need to be addressed. Some kind of information about the network topology is usually needed. A great number of proposals have been presented in the context of WSANs [85][86][87][88]. Reliability, routing maintenance, latency and energy efficiency are some of the requirements that can be measured/specified at this level.

In [89], the SAR algorithm is used to select which path to use for sending information to a sink node. Multiple paths from each node towards the sink node are assumed to be available. A QoS metric is computed for each packet routed through the network. This metric takes into account the energy cost, delay of every path and the priority of the packet. The final objective of the SAR algorithm is therefore, to minimize the average weighted QoS metric throughout the lifetime of the network.

SPEED [90] is a stateless, localized routing algorithm for WSNs that provides real-time communication. By assuming nodes are aware of their location, routing can be done by selecting at each step, nodes which are closer to the destination than the actual one. Relay speed is calculated from the distances between nodes and the delay between them, and used to maintain a routing speed. Additional modules are provided to reduce or divert traffic when congestion occurs and to try to balance the network load. Because SPEED only maintains immediate neighbor information, a beaconing module is used to exchange neighbor locations. SPEED, however, does not provide any guarantee in the reliability domain.

MMSPEED [91] borrows some ideas from SPEED and extends the protocol to deal with reliability. The single network-wide speed approach taken by SPEED is generalized by replicating it and using multiple SPEED layers that run independently of each other. Packets are classified according to their speed classes and placed in priority queues. This way different traffic classes can be treated according to the timeliness' requirements. Reliability is probabilistically guaranteed taking into account packet loss percentage and hop count to the destination node. However, MMSPEED needs the MAC layer to provide advanced functions such as prioritized access to the medium or reliable multicast delivery to multiple neighbors.

ReInForM [92] is a routing protocol which provides reliability by providing a multiple-

path routing scheme. Different priority levels are used to classify packets. Each priority level maps to a desired reliability in data delivery. Depending on the criticality of the data, multiple copies of the message are sent over multiple-paths towards the sink. ReInForM routing only uses local information and does not require any data caching at any node. Routing is carried out based on the information about error rates (typically provided by the MAC layer) and number of hops to the destination.

Energy-Aware QoS Routing [93][94] proposes a routing protocol that handles delay constraints and energy consumption but also ensures that best-effort traffic is correctly managed. The sensor network is assumed to be grouped in clusters, each of them managed by a gateway. All gateways can directly communicate with a command node that controls the whole network. Two priority queues are used at each node to classify and process priority and non priority data. An additional bandwidth ratio r is used to control the bandwidth dedicated to both real-time and non real-time traffic. To route real-time traffic a cost function is associated with each link. This cost function takes into account distance between nodes, node energy, error rate and propagation factors. From the link cost function least cost paths are calculated. A K least cost path algorithm is used to find a set of candidate routes. Paths in this set are checked against the end-to-end constraints and the one that provides maximum throughput is used.

DAPR [95] makes use of an “application cost” to route packets. The application cost measures the importance of individual sensors to the sensing application. This indicator takes into account the energy of the node and the contribution it makes to the sensing tasks relative to their neighbors. In DAPR, time is divided into three different phases that form a round. Each round is signaled by the data sink by means of a round start message. The first phase is called the route discovery phase where the path to route the packets is determined using the application cost function. The second phase (role discovery phase) is used to decide whether nodes need to become inactive in the next round. That happens when the area of interest is entirely covered by neighbor sensors that are going to stay active in the next round. This synchronization is managed by means of signal beacons. Finally, the last phase is where the actual routing and normal sensor operations take place.

SOMDV-R [96] is based on the previously proposed AOMDV protocol but extended to support reliable data forwarding. The SOMDV-R operation is composed of four phases. In the first one, called local connectivity maintenance, each node periodically communicates

with its neighbors to know the qualities of their links. Also, in this phase, topology changes are detected. In the route information gathering phase, a source node initiates a discovery route to destination, either because it does not already have one or because the one it has does not meet the QoS requirements. During this phase, all nodes involved update their information about the network state. In the third phase, that is route/path update and maintenance, a route is deleted if some of the nodes involved in it fail and mechanisms to repair a local route or to reroute a packet are used if necessary. Finally, the last phase is where the actual data routing takes place by using all the information gathered in previous steps. If there is a path that meets all the QoS requirements then the packet is sent over it. If not, multiple paths are used in order to meet the requirements.

In [97] a hierarchical routing algorithm for oil, gas and water pipelines is presented. These scenarios can all be considered critical infrastructures. The routing algorithm relies on a set of different nodes arranged in a hierarchy. Basic Sensor Nodes collect the information and relay it to the Data Relay Nodes. Data Relay Nodes, then relay the information to the Data Discharge Nodes which will eventually discharge the data to the Network Control Center. The routing relies on an hierarchical addressing scheme in order to calculate the next hop at each routing stage. This routing addressing scheme however is statically initialized. Moreover, the use of Data Relay Nodes and Data Discharge Nodes makes it vulnerable to attacks that would have a considerable impact on a set of Basic Sensor Nodes.

Dwarf [98] is a routing algorithm that supports QoS. More specifically, it provides reliable data delivery, low latency and low energy consumption. Nodes are organized in a hierarchy based on their distance from the sink. Each node is initially asleep and periodically wakes up to transmit data to the sink through the neighbor that wakes up next and is nearer to the sink. The MAC is assumed to be able to provide information about the wake-up schedule of each neighbor. In order to discover the neighbor status, a periodical maintenance protocol is carried where nodes exchange their view of the network. Reliability is achieved by means of a retransmission scheme.

Table 3.4 shows the main features of the routing protocols presented in this section. The “QoS parameters” column shows the QoS characteristics that the routing algorithm takes into account when routing packets.

Due to the special CIP application needs, the routing algorithm needs to be carefully chosen. Many proposed routing algorithms assume node location awareness such as SPEED,

Name	RT Type	QoS parameters								Energy efficiency	Scalability
		EN	D	R	B	ER	P	C	M		
SAR	Soft RT	X								N/A	Limited
SPEED	Soft RT		X							N/A	Good
MMSPEED	Soft RT		X	X						N/A	Good
REINFORM	Soft RT			X						Low	Good
Energy-Aware QoS Routing [93]	Soft RT	X	X			X				High	Limited
DAPR	Soft RT	X						X		Good	Limited
SOMDV-R	Soft RT			X						Moderate	Limited
Hierarchical routing for pipelines	Soft RT			X						N/A	Good
Dwarf	Soft RT	X	X	X						High	Moderate

Table 3.4: WSAN Routing protocols summary (EN: energy consumption; D: delay; R: reliability; B: bandwidth; ER: error rate; P: priority; C: coverage; M: mobility)

MMSPEED or Delay-aware Reliable event reporting framework. This is generally not desirable for CIP applications since the system in charge of acquiring the location can be subject to attack. For example the GPS system used by a routing algorithm can be altered and so, the whole routing algorithm (depending on the case) can be damaged.

In addition, reliability support is a must as the data handled by systems of this kind is critical. REINFORM and SOMDV-R and the routing protocol presented in Delay-aware Reliable event reporting frameworks make use of multiple paths to greatly improve the reliability. Although this method if not used properly can lead to great energy consumption its use seems feasible for the CIP. The approach taken by Energy-Aware QoS Routing is also interesting since critical data is given more priority but at the same time data with best-effort requirements is eventually handled. However, the topology assumptions they make need to be carefully studied, as they can also be used to attack important nodes in the network. For example, in cluster-based topologies the failure of a cluster-head affects the whole cluster. In the same way, in [97] an attack to a Data Relay Node would make all Basis Sensor Nodes reporting to it unable to transmit any information to the sink. In that sense, this routing algorithm although currently being applied to critical infrastructures does not effectively provide the QoS required in such scenarios. The reasons given, do not mean to imply that hierarchical architectures should not be used in CIP systems. They provide great flexility and effective network management but where that approach is adopted, mechanisms to deal

with cluster-head failures, energy consumption, etc need to be used.

Dwarf duty cycling mechanisms and hierarchical organization of the network are interesting. However, the algorithm does not handle different traffic classes and assumes that the neighbor wake-up schedules are known by the MAC protocol. Delay-aware Reliable event reporting framework assumes that some information is known about the packet queue state of neighboring sensors. This information is used to select neighbors that are not busy transmitting. However, it is not clear whether this information is really up-to-date and can be used for that purpose. WSANs are highly dynamic systems and the state of a queue may not always be predictable.

Finally, in general, all proposals concentrate on various aspects of QoS but there is still a need to devise new QoS-aware protocols that take into account the main QoS requirements in the CIP context such as dependability, real-time and energy efficiency at the same time.

3.2.3 Hybrid protocols

To provide QoS the WSAN resources need to be managed in the most efficient way. In order to maximize the use of resources, cross-layer interactions have been used to allow two communication layers to interact with each other and exchange information. In this way, for example, control and network state information can be used to optimize and offer a better performance. Typical cross-layer interactions communicate network, MAC layers and sometimes the application layer. Hybrid protocols refer to the protocols that involve more than one typical communication layer and that use a cross-layer design to improve the efficiency. For example, MAC information about connectivity can be used to improve routing at network layer.

A couple of hybrid protocols that deal with QoS are covered in this section and their suitability for CIP applications is discussed.

RAP [99] presents a real-time communication architecture for large-scale sensor networks that provides the MAC layer, network layer, transport layer and application layer. RAP needs to run over a MAC layer that supports distributed prioritization. For this reason, an extension of the IEEE 802.11 wireless LAN protocol has been chosen as the MAC layer of the tested framework. The network layer is assumed to be location-aware. The location is used to route the information towards the destination node in a greedy way. To meet QoS constraints specified in the communication between nodes, a velocity monotonic

Name	MAC	Routing	Transport layer	QoS parameters	Scalability
RAP	prioritized MAC	Location-aware, routing based on velocity	Location-based	Deadline, period	Moderate
LEACH	CDMA-based	TDMA-scheduled from cluster-head to sink	N/A	Energy consumption	Moderate

Table 3.5: WSAN QoS hybrid protocols summary

scheduling is proposed. This approach routes packets taking into account the requested velocity (velocity at which packets need to move to meet the deadline constraint). Priorities are assigned according to the resulting requested velocity. The transport layer uses location rather than address to deliver data. This means that nodes are not directly selected by a unique identifier, but selected depending on the location they have. Finally, a high level API is provided that allows developers to submit queries or register to certain events.

LEACH [100] is a clustering algorithm that aims at minimizing energy consumption by evenly distributing the energy load among the sensors in the network. The network is divided into different cluster-heads. The position of cluster-head rotates between the different sensors in the cluster so that the energy consumption is balanced. Each node determines whether or not it wants to become the cluster-head independently of other nodes based on probability. Once clusters have been formed, the cluster-head creates a TDMA schedule telling each node when it can transmit. Nodes which do not need to transmit can be turned off. The article proposes the use of CDMA techniques to avoid interference between clusters.

Cross-layering can be advantageous as performance can be maximized. The main disadvantage of using it is the lack of flexibility due to the coupling between layers. If changes in one layer need to be made, all the other layers with which the former has cross-layer interactions also need to be adapted. However, since in CIP applications, dependability and real-time constraints are of crucial importance we think that the use of a cross-layer design is justified. By developing a specific network and MAC layer and ensuring that the interaction between them is optimum a better QoS can be provided. A drawback of this is that the use of a standard can somehow be contradictory with a cross-layer design, as functionality that can be accessed in a particular layer is determined by the standard itself. In this matter, many existing cross-layer designs opt for using standards and extending the functionality they offer to allow cross-layer interactions.

Table 3.5 shows the main features of the hybrid protocols and frameworks surveyed.

3.3 SCADA and the integration of WSANs

SCADA systems are widely used nowadays to monitor and control dispersed hardware components in industrial settings such as power plants, gas or water treatment [101]. Often, they are used in critical infrastructures where security is a vital factor for the environment or human health. Due to this, they have to satisfy strict regulatory standards [102]. Traditionally, SCADA systems have been deployed in a monolithic way with a central datacenter and where a large amount of wiring is required to connect the different hardware elements with the datacenter. SCADAs normally comprise isolated and proprietary hardware, software and protocols. Due to the restricted knowledge about these elements, previous security efforts were minimal and focused primarily on physical measures [103]. Recently, the use of WSAN technology has been acknowledged as promising for the CIP field. We believe WSANs find in SCADAs, a mature technology that can be used to collect the information sensed by the network, to analyze it and to show it to users in a elegant way. WSANs exhibit some unique characteristics that are interesting in the context of SCADA systems:

- Dynamism: WSANs are not static networks and allow connecting new nodes to the network and disconnecting or deactivating nodes deployed. The lack of wiring between the devices allows node mobility and eases the task of changing the network topology.

- Retrofit: SCADA systems are usually deployed in a large scenario. That means that updating and changing the architecture once it has already been deployed is not simple. Reprogramming techniques can be used, but since the location of the devices is static and normally those devices require wiring, the updates are costly. On the other hand, diverse reprogramming techniques have been proposed in the context of WSANs that can be used to update the behavior of the monitoring application deployed in the WSAN.

- Ease of installation: WSANs can be easily deployed in the electrical power grid scenario since a separate energy source is not needed to power the nodes. Energy harvesting from the power grid can be done to feed the sensors. Furthermore, the wireless capabilities also help to reduce the cost of deploying a monitoring network.

- Redundancy: WSANs can be composed of a large number of nodes that cooperate to monitor a scenario. This high density of devices can be used to provide fault tolerance and QoS by exploiting node redundancy.

In Chapter 7 we propose a SCADA/WSAN architecture to deal with the integration and accessability problem. The numerous advantages of merging SCADA systems and WSAN with the internet have been recognized in the literature [104] such as wide area connectivity and pervasive access, redundancy, large addressing range, etc. The main advantage of this approach is the security. By exposing a SCADA system to the internet it becomes a potential subject of attack by hackers. However, we believe that if good security measures are used the potential benefits they provide justify their use. Much work exists, related to this same subject. In [105] the authors propose replacing existing SCADA systems with a WSN. They share some of our points of view: advantages of WSN, simplicity, static SCADAs. The approaches presented in [106] and [107] both have case studies more similar to the one presented in our proposal. They present the integration of WSAN in wind power plants in the first paper and their integration on power grids in the second. Our approach is different in the sense that we use open technologies, which provide Internet access to the monitored system. In addition we consider security issues, which are critical in these systems. Along the same lines, the work presented in [108] presents a framework solution which focuses more on security for SCADA systems and from a high level perspective. In our solution, security is taken into account both at the service level and in the SCADA.

4

PS-QUASAR middleware

This chapter deals with the three main aspects described in Chapter 1. The first one is the need to provide a higher level of abstraction to simplify the development of WSAN applications. Many different programming models, as explained in Section 3.1, have been proposed. In this sense, we believe that the publish/subscribe model is a feasible programming abstraction for WSANs and one that matches the reporting paradigm that a WSAN usually has, that is, a large number of nodes report their readings to a small number of data consumers. For the developers, it greatly simplifies the task of programming since they do not have to worry about low level communication primitives and can focus on the program logic rather than on how to distribute the data it generates.

The second issue is that QoS support is essential for most modern applications, in particular for reliability, therefore the proposed middleware layer must be able to handle different QoS requirements.

The final issue tackles the dynamicity of WSANs. WSANs are usually deployed in changing and adverse environments and therefore routing protocols should be able to han-

dle new node additions or node failures. This, for example, means that the communication scheme should not only rely on a node's physical addresses, because available nodes' addresses can change over time if new nodes are added or existing ones fail. Also, because the publish/subscribe programming abstraction requires specific routing requirements such as the use of a multicast mechanism or to keep track of publishers and subscribers locations, the routing protocol must be able to provide these features.

Let us take an example application scenario of a WSAN for monitoring and controlling an electrical substation. The sensors are attached to power transformers and power lines to measure temperature, current and component status. Readings outside of the normal range are notified as warnings and alarms and need to be reliably sent to a SCADA with high priority. In addition, several movement detectors are deployed in the electrical substation to detect unauthorized intrusions. Finally, a couple of cameras and luminous and acoustic alarms have been deployed in the scenario. These cameras and alarms are activated when an alarm notification, warning notification or intrusion are detected. The WSAN in this scenario needs to cope with sensors and actors, priority messages, reliable transmission and multiple subscribers (alarms, SCADA and cameras). This application scenario is based on a real demonstrator presented in the WSAN4CIP project [64] in which we participated. In fact the same case study is used in Section 7 to illustrate the architecture to integrate SCADAs and WSANs.

Taking these requirements into account and the application scenario as the motivation, this chapter proposes a publish/subscribe middleware based on a simple publish/subscribe model suitable for WSANs called PS-QUASAR that is able to provide QoS support. The routing protocol that comes with PS-QUASAR (referred to as the PS-QUASAR routing protocol in the following chapters) is fully distributed and does not assume any reporting pattern. PS-QUASAR uses a publish/subscribe directed acyclic graph based routing protocol that supports a many-to-many communication and can handle priority, deadline and reliability requirements in the communication between nodes.

There are publish/subscribe protocols which have already been proposed in the context of WSANs and these are covered in Section 3.1.7, illustrating their differences to PS-QUASAR. The main contributions of our proposal are as follows: 1) A middleware that supports QoS and which provides a simple programming model based on the publish/subscribe paradigm 2) a routing protocol supporting a publish/subscribe paradigm in a

fully distributed way, based on directed acyclic graphs.

The rest of the chapter is structured as follows. Section 4.1 describes the problem formulation and the scenario assumptions. The PS-QUASAR middleware and implementation notes are presented in Section 4.2. The performance and evaluation tests are described and analyzed in Section 4.3.

4.1 Problem formulation, assumptions

The network is assumed to be composed of an homogeneous set of nodes with equal transmission range. Nodes can interfere with the communications of other nodes if communication is simultaneous. The following entities are possible in the network: 1) publishers (data producers) and/or subscriber (data consumers) 2) relay nodes which are neither publisher nor subscribers. Nodes are not aware of their position or the network topology and the reporting pattern is unknown, i.e., nodes can freely send information when they want. The goal is to route information from publishers to subscribers that have declared their interest in the same topic and by taking into account the QoS requirements.

Let the network graph be $G = (V, E)$ where V is the set of nodes and E the set of edges. The cardinality of each set is specified as $n =\mid V \mid$ and $m =\mid E \mid$ respectively. Let s_i be node i in V and e_{ij} the edge that goes from s_i to s_j where $0 \leq i, j \leq n$. Let N_i be the set of neighbors adjacent to s_i and $t_i =\mid N_i \mid$. We consider that two nodes are adjacent if they can directly communicate with each other. Let Pub_t and Sub_t, respectively be all the publishers and subscribers of topic t. $D(s_i, s)$ gives the minimum known distance at a given time between node s_i and subscriber s measured in number of hops.

4.2 PS-QUASAR middleware

PS-QUASAR is a lightweight middleware that provides a topic-based publish/subscribe scheme with an event/listener abstraction over the traditional message passing abstraction provided by all current WSAN platforms. The PS-QUASAR routing protocol is especially tailored to the publish/subscribe requirements. In other words, it connects multiple publishers to subscribers in a transparent way for the programmer and at the same time ensures that the QoS requirements are met.

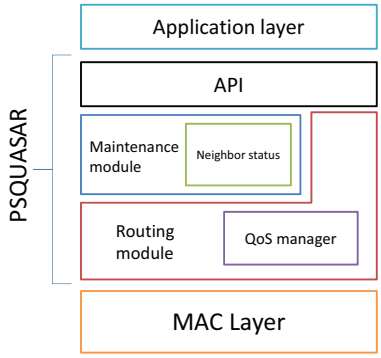


Figure 4.1: PS-QUASAR module diagram

The PS-QUASAR middleware is composed of three different modules: PS-QUASAR Maintenance Protocol, PS-QUASAR Routing Module and the API. Figure 4.1 shows the PS-QUASAR module diagram and how the different modules connect to each other. The maintenance protocol is in charge of creating the links between neighbor nodes and discovering subscribers and publishers. The routing module carries out the actual routing process based on the information collected by the former protocol. The API, on the other hand, provides a set of methods for developers to make use of the publish/subscribe programming model offered by PS-QUASAR. The details on these three components are presented in the following sections. Section 4.2.1 presents an overview of the publish/subscribe programming model provided by PS-QUASAR. Section 4.2.2 presents the maintenance protocol and unsubscription mechanisms used by the PS-QUASAR routing module. The routing protocol is explained in Section 4.2.3. Finally, the QoS support mechanisms used are detailed in Section 4.2.4.

4.2.1 Publish/subscribe programming model

PS-QUASAR provides a lightweight topic-based publish/subscribe model suitable for wireless sensor devices, which are constrained devices in terms of size, cost, memory and bandwidth. Topic-based publish/subscribe means that the matching process between publishers and subscribers is done entirely on the basis of a topic name. A topic is simply a keyword that identifies the subject of the information generated by a publisher or the subject of the information that a subscriber wants to receive. In PS-QUASAR the topic name is encoded in a 16-bit integer for reasons of efficiency. Each piece of information generated by

```
// SUBSCRIBER
ps_subscribe(char* topic_name, listener_function listener);
ps_unsubscribe(topic_type topic_id);
ps_unsubscribe(topic_type topic_id, listener_function listener);

// PUBLISHER
ps_publish(char* topic_name, char* data,
           unsigned short data_size, struct QoS_policy qos_policy );

Qos_policy{
    // Deadline expressed in ms
    Number deadline;
    // Indicates the number of retransmissions
    Number reliability;
    Number priority;
}

// OTHERS
ps_notify_listener(char *topic_name, char* data_to_listener, rimeaddr_t* addr);
ps_print_status();
```

Figure 4.2: PS-QUASAR programming model API

publishers is associated with a 16-bit integer (chosen by the developer and specified in the ps_publish primitive). This information will be automatically delivered to nodes which have declared their interest in the same 16-bit integer (by means of the ps_subscribe primitive). The publish/subscribe programming model provided by PS-QUASAR is really simple and easy to use. The simplicity of the model helps developers to implement WSAN applications without worrying about common low-level issues such as data packet encoding/decoding, message handling, etc.

The publish/subscribe programming model API is shown in Figure 4.2. The primitives are classified depending on whether they are called from the publisher or the subscriber side. The proposed publish/subscribe programming model is based on two different mechanisms: publish/subscriber primitives and listeners. The publish/subscribe primitives allow information to be transparently sent from publishers to subscribers. These two entities can be located in different nodes or in the same one. By doing this we use an homogeneous publish/subscribe approach that is used for all communications that are carried out in the network, either in a local or remote way. In the proposed publish/subscribe model, the QoS requirements are only specified on the publisher's side. This simplifies the task of

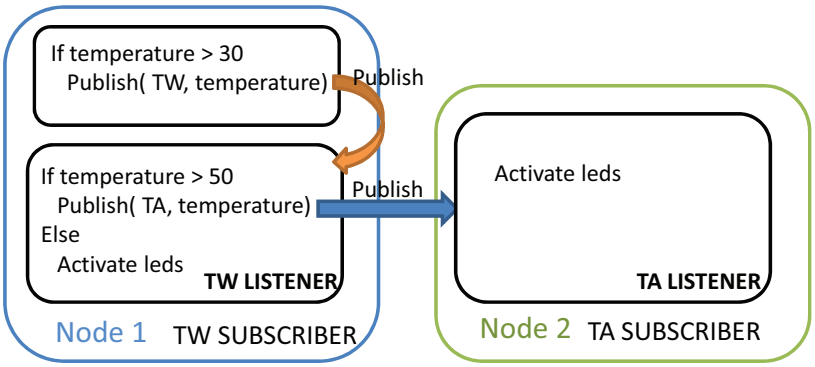


Figure 4.3: PS-QUASAR programming model example

delivering information and avoids time-consuming QoS-matching algorithms. In DDS, for example, QoS is specified on both sides and only when the QoS policies are compatible does the communication take place. This however requires an additional matching algorithm or must rely on a central server that carries out this task. The QoS parameters offered are deadline, reliability and priority. Deadline is expressed in ms, reliability in number of node-to-node retransmissions. Priority is expressed as a number, the higher it is the more priority is given to the packet. More details about how these QoS parameters are handled can be found in Section 4.2.4. Listeners are functions that are executed whenever a message is received by a subscriber. Only subscribers can make use of listeners to process received data. Listeners are specified as a parameter in the ps_subscribe method. Listeners receive two parameters: the ID of the node sending the information and the data itself. In addition, the developer needs to specify the behavior of the main function. This function is called once, upon initialization. From it, subscriptions are chosen (ps_subscribe) and publish methods can also be called. For example if periodical reporting is needed, periodical timers are set in the main function.

To sum up, the PS-QUASAR programming model is composed of publish/subscribe primitives and listeners. Publish/subscribe primitives are used to communicate nodes and functions in the same node or in remote nodes whereas listeners take action whenever information is received at a node. Figure 4.3 illustrates this entire process by means of a simple example. In the example, node 1 publishes a temperature warning notification(topic TW). Because only the local node is subscribed to the TW topic no communication is carried out with remote nodes. A local listener processes the data and depending on its value publishes

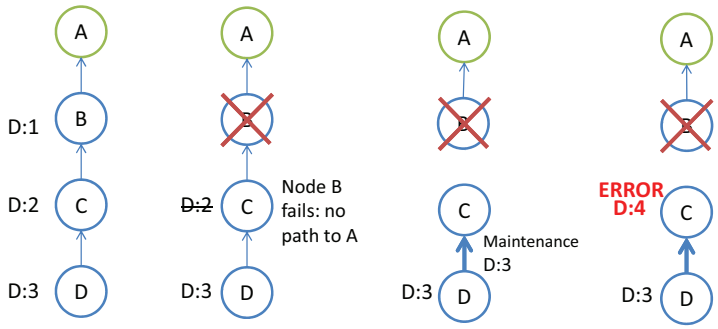


Figure 4.4: Node failure issue in common tree-based maintenance

the reading to all nodes subscribed to the temperature alarm topic (TA topic), which is node 2 in the example.

4.2.2 Maintenance protocol

Tree-based routing protocols usually require a maintenance protocol that allows each node to find its neighbors and coordinate them to form a tree. One of the classic algorithms to form a routing tree in a distributed manner is the Bellman-Ford algorithm [109]. The algorithm maintains a table at each node with the best known distance to each destination (subscriber) and where it has to relay the information to get there. This distance is iteratively updated by exchanging information with its neighbors. Eventually, each node in the network is able to find the shortest distance to the destination. The distance is typically measured as the number of hops towards the destination.

One identified issue in the Bellman-Ford algorithm is the count-to-infinity problem shown in Figure 4.4 and explained in [109]. Let us imagine we have a network of 4 nodes. The root is supposed to be node A. After the maintenance protocol is carried out all nodes know how to relay the information to A. At a certain point node B fails and because of the timestamps, node C is aware that no path is available to node A. Before node D is aware of this new situation it sends a maintenance packet stating that node A is at distance 3. Node C will receive this maintenance packet and it will believe that there is a new path towards A if information is relayed through D. If there are more nodes in the network this error will cascade through them and the network will become unusable.

Section 4.2.2.1 covers the PS-QUASAR maintenance protocol based on the Bellman-

Ford algorithm and Section 4.2.2.2 the unsubscription/failure recovery mechanism that also deals with count-to-infinity problem. The PS-QUASAR maintenance and unsubscription protocol described in both sections is depicted in Algorithm 1.

4.2.2.1 PS-QUASAR Maintenance

The PS-QUASAR routing protocol is based on a tree-based routing protocol and as such requires a maintenance protocol. The proposed maintenance protocol is a modified version of Bellman-Ford algorithm. The maintenance protocol is based on periodical beaconing to exchange node information (e.g. remaining energy) and subscriber information. Node information is used to meet QoS requirements.

All nodes periodically broadcast a maintenance packet with information about themselves and also with information about the subscribers that are accessible from the current node. In every node, information about the subscribers found so far and information about the neighbors are kept in a local table. This table keeps track of all the subscribers discovered so far and all the possible neighbours through which the node can relay the information to make the packet reach the subscriber. Because of this, PS-QUASAR routing protocol is not strictly a tree-based routing protocol but rather a directed acyclic graph based protocol. All possible next-hop that are part of minimum distance paths are stored and are taken into account during the routing process. If one of the paths with minimum distance towards the destination fails the others are still available and the routing protocol will continue to operate normally. Only when all paths with minimum distance fail will the unsubscription/failure protocol presented in Section 4.2.2.2 be executed.

Two facts can be deduced from the proposed maintenance protocol. The first is that all nodes in the network are aware of the number of subscribers and how to reach them and the second is that multiple paths towards each subscriber are identified. The first fact means that in PS-QUASAR all nodes in the network can act as publishers without having to notify of this fact in advance (unlike traditional publish/subscribe systems). The second means that PS-QUASAR can select the most suitable path towards a subscriber based on the state of the network. Figure 4.5 shows the information exchange by the maintenance protocol. It includes the distance to each of the subscribers, information about each of the neighbors (such as remaining energy) and unsubscription information.

Algorithm 1 PS-QUASAR Maintenance protocol. Node s_i

loop
  for all active unsubscription of subscriber a from neighbor n **do**
    if $phase(n, a) == 1$ **then**
      Add to notify packet: a is no longer accessible from s_i
    end if
  end for
  for all subscriber b accessible from node s_i **do**
    Calculate $D(s_{i,}, b)$ from the information stored in the neighbor table (nodes in N_i)
    Add to notify packet: subscriber b is at distance $D(s_i, b)$ from node s_i
  end for
  Add to notify packet: information about s_i such as $E(s_i)$, ...
  Broadcast notify packet
  wait MAINTENANCE_PERIOD
end loop
while reception of notify packet p from node s_j **do**
  for all unsub./failure of subscriber a accessible from neighbor s_j in p **do**
    if subscriber a is only accessible from s_j with min. distance **then**
      delete from neighbor table: a is accessible
      $phase(s_j, a) = 1$
      start timer to set $phase(s_j, a) = 2$ after time t_1
      start timer to end unsub. protocol for subscriber a and neighbor s_j after time $t_1 + t_2$
    end if
  end for
  for all subscriber b notified in p **do**
    if $phase(s_j, b) = 1 \;||\; phase(s_j, b) = 2$ **then**
      ignore notification (the subscriber is unsubscribing)
    else
      update distance to b as $D(s_j, b)$+1 if it is nearer the one already stored in the local tables
    end if
  end for
  update information about the neighbor s_j: $E(s_j)$, ...
  update s_j timestamp
end while

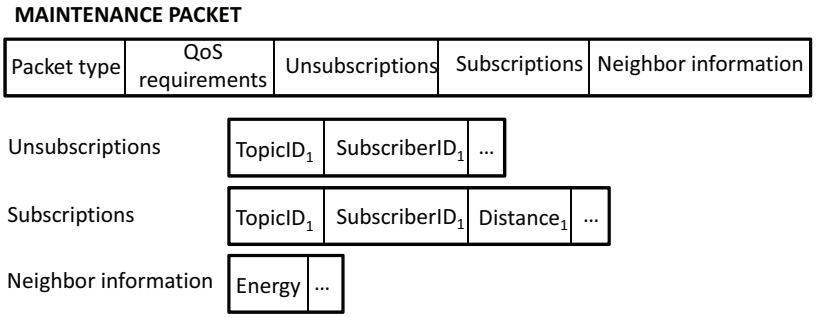


Figure 4.5: Maintenance packet fields

4.2.2.2 Unsubscription and fault tolerance

The unsubscription algorithm uses the periodic maintenance mechanism to carry out unsubscription and node failure notification. The same maintenance packet used in the protocol explained in Section 4.2.2 is also used to notify of unsubscriptions and node failures. If any unsubscription is issued then the maintenance packet is also used for a number of periods to notify neighbors that a subscriber is no longer available.

In addition, the proposed algorithm has identified and tackled the count-to-infinity problem. This behavior is caused because nodes update their tables with information, that is not up-to-date which can cause an incoherent state that prevents routing to be carried out properly. Whenever an unsubscription or a node failure takes place the neighbors will be aware of the situation because of the timestamp mechanism. Each neighbor will detect the subscriptions involved in the unsubscription/node failure. Then a two phase unsubscription process will start for that particular subscription. One phase runs after the other and each of the phases lasts for a predefined length of time that is measured using timers. Different unsubscriptions can be executed at the same time in a node and each of them can be in a different phase.

When an unsubscription of subscriber u starts at node n:

1. During the first phase node n will periodically broadcast that u is unsubscribing. Information received at node n about subscriber u is discarded during this phase. In other words, phase one indicates that a subscriber is currently unsubscribing and therefore the information it sends may not be up-to-date. Packets receiving the unsubscription of subscriber u from node n will start the unsubscription protocol for

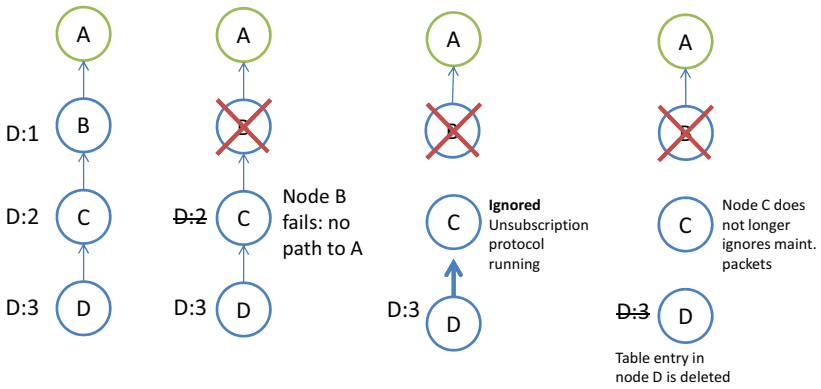


Figure 4.6: PS-QUASAR maintenance example

that particular subscriber. This way the unsubscription information propagates across the network.

2. This phase is similar to phase 1 but no broadcast messages are sent informing about the unsubscriptions/node failures. Phase 2 gives time for neighbors of node n to delete the corresponding entries about subscriber u from the tables before trusting in the content of the corresponding maintenance packet sent by node n. This way the count-to-infinity problem is mitigated. For this to be done, the duration of phase two must be equal to or higher than the time defined in the timestamp to delete old information about nodes.

Phases are symbolically marked as $phase(s_i, a)$ in Algorithm 1. Finally, we note that the maintenance mechanism is used to carry out unsubscription so no additional messages are needed to add the modifications proposed.

Figure 4.6 shows an example of the same situation shown in Figure 4.4 with our proposal. This time the maintenance packet that node C receives from node D is ignored because node C is running the unsubscription protocol for that particular topic and subscriber.

4.2.3 Routing protocol

The routing protocol is covered in this section. First the concept of weight function is introduced in Section 4.2.3.1. Then, the multicast mechanism and the PS-QUASAR routing algorithm is explained in Section 4.2.3.2.

4.2.3.1 PS-QUASAR Weight function

The PS-QUASAR routing protocol uses a weight function to evaluate each of the neighbor nodes to which it can relay data packets. The PS-QUASAR weight function that evaluates node s_i to calculate the importance of link e_{ij} is defined as

$$W_i(s_j) = f_f(neighbor\_status) * f_w(neighbor\_status) \tag{4.1}$$

where f_w is a function that evaluates the importance of the path towards a subscriber based on the information it has about the network and about the neighbor. By defining function f_w so that it takes into account different parameters such as LQI (link quality indicator), remaining energy, traffic, etc, routing decisions can be customized to tailor the protocol behavior to the need of the users. Finally, the weight function is controlled by an additional function f_f. This allows the protocol to discard certain nodes which do not want to be considered as relay nodes. Basically when a node wants to be discarded as a relay node, f_f must be equal to 0. Since the weight function expresses the probability of a node to be chosen as the next relay node, a value of 0 indicates that the node is never going to be chosen in the routing process. The weight function is the mathematical tool that shapes the behavior of the routing module.

For example let us consider that we want the routing algorithm to give priority to paths with a high remaining energy and with a high LQI. One possible solution is to define the weight function as in Formula 4.2. E_m and L_m are the normalized value of the remaining energy of a node and the LQI value of the communication link with that node respectively. c_1 and c_2 values are used to further control the importance of the energy and LQI parameters.

Additionally, only nodes whose energy and LQI are higher than a predefined threshold will be taken into account in the routing process.

$$\begin{aligned} f_f(s_j) &= \begin{cases} 0 & \text{if } E(s_j) < THRESHOLD1 \bigvee L(s_j) < THRESHOLD2 \\ 1 & \text{otherwise} \end{cases} \\ f_w(neighbor\_status) &= c_1 * E_m(s_j, neighbor\_status) + \\ & c_2 * L_m(s_j, neighbor\_status) \\ W_i(s_j) &= f_f(s_j) * f_w(neighbor\_status) \end{aligned} \tag{4.2}$$

4.2.3.2 PS-QUASAR routing

PS-QUASAR does not cache information. The information is sent as soon as it is produced by means of a directed acyclic graph based routing protocol. The PS-QUASAR routing protocol extends the concept of a simple routing tree to make use not only of one path towards each subscriber but to use as many paths as possible. To achieve this, nodes establish links with each other to form a directed acyclic graph for each of the publishers in the network. Although this implies that more memory is necessary to keep the information about all adjacent nodes, the number of neighbors a sensor node has, even in dense networks, is relatively small. This extra memory can be handled perfectly well by current sensor nodes.

The PS-QUASAR routing protocol dynamically adapts its behavior to the neighbor status to route the packets using different paths and allows the information to be transmitted with certain QoS policies. The use of multiple paths towards the subscriber node improves the energy efficiency of the algorithm. As explained in Section 4.2.2, the maintenance protocol identifies the possible paths towards each of the subscribers. The routing protocol uses this information to send data packet associated with a topic to all subscribers of that topic in the network.

PS-QUASAR proposes a publish/subscribe programming model that allows multiple subscribers to declare their interest in the same topic. Because of that, data associated with a topic may need to be delivered to more than one node in the network. One option would be to send a packet for each of the subscribers in the network. This however, generates unnecessary traffic as the same information is duplicated. To optimize this type of communication PS-QUASAR proposes a multicast mechanism.In general terms, packets are not duplicated whenever possible unless the network topology requires it.

The routing module takes the information that the maintenance protocol collects and reuses paths whenever possible when more than one subscriber of a topic is detected. If subscribers cannot be reached by relaying to the same direction then packets are duplicated. In order to control duplicate packets a packet field called “Already notified” (Figure 4.7) is used to tell the nodes the list of subscribers that do not need to be notified to the nodes. This field contains a list of all subscriber ids that are being notified by other packet duplicates or that have already being notified. The use of that packet field and the algorithm itself is explained by means of an example in Figure 4.8.

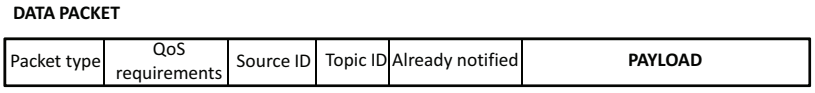
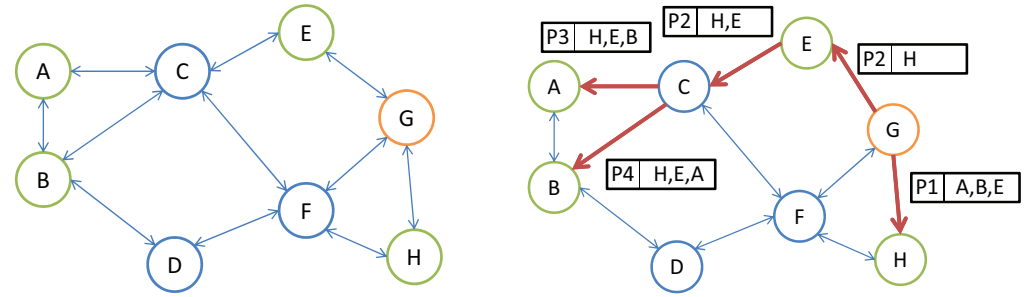


Figure 4.7: Data packet fields



(a) Example scenario. Node G is the publisher and nodes A, B, E and H are the subscribers

(b) Packet duplication is not done unless required by the network topology

Figure 4.8: PS-QUASAR multicast mechanism

In the example we have four subscribers (nodes A,B,E,H) and one publisher (node G). In Figure 4.8(b) publisher G publishes data to subscribers A,B,E and H. For each of the messages sent, the content of the "Already notified" field of each packet is shown. The decision of which path to use is based entirely on the information collected by the maintenance protocol. More specifically, distances are used to detect paths that take us one hop closer to the maximum number of subscribers possible. The algorithm is repeated in each of the relay nodes until all corresponding subscribers are attended to by the smallest number of packets. In the example node G detects that by relaying through node E it is possible to reach nodes E, A, B. Another packet is sent to node H to reach the remaining subscriber. The "Already notified" field of the packet sent from G to E specifies that this packet does not need to reach H because there is another packet that is in charge of doing so. In this way, the field is updated every time a packet needs to be duplicated or a subscriber is notified. This field is necessary for packets to keep control of the multicast process.

The pseudocode behavior of the PS-QUASAR routing protocol is detailed in Algorithm 2.

Every time a data packet is received the "Already notified" field of the packet is compared to the subscriber tables in the node to check whether there are still any subscribers

Algorithm 2 PS-QUASAR routing module pseudocode. Node s_i

while reception of data packet p of topic t **do**
calculate a set of nodes $H \subseteq N_i$ so that
1. by relaying a packet through every node $s_h \in H$ all subscribers not attended yet in Sub_t can be reached with minimum distance
2. $\mid H \mid$ is minimum
3. $\mid H \cap L \mid = \varnothing$ where L are the nodes specified in the already notified section of packet p
4. If there is more than one possible option to reach a subscriber then the path whose weight function is bigger is chosen

update deadline packet field and end if it has expired
for all $s_h \in H$ **do**
update status of "Already notified" field of packet p
if reliability is enabled **then**
relay data packet p to node s_h in a reliable way
else
relay data packet p to node s_h
end if
end for
end while

that have not been notified. If there are any then the corresponding paths are chosen so that all remaining subscribers are covered, the number of packet duplicates is minimum and also whose weight function is the biggest. For each of the copies of the packet that are relayed, the "Already notified" field needs to be updated to include all subscribers covered by other packet duplicates.

4.2.4 QoS

This section summarizes the mechanisms used to deal with the QoS requirements specified by means of the publish/subscribe programming model detailed in Section 4.2.1. These are: reliability, priority and deadline.

In our protocol, reliability deals with increasing the probability with which packets are transmitted whereas priority finds short paths towards the destination based on the importance of the packets. Reliability is achieved using node-to-node retransmission. The number of retransmissions that is made on a node-to-node basis can be specified in the publish primitive shown in Figure 4.2. Priority is handled by means of queues where incoming and outgoing packets are ordered according to their importance. Priority is specified as an integer number that is stored inside each packet.

Finally, PS-QUASAR provides a deadline mechanism that allows packets to be discarded if their deadline has expired before reaching destination. The user can specify a deadline that expresses the maximum time allowed for the packet to reach the destination. In PS-QUASAR, the deadline, if specified, is stored in a field in each data packet and updated everytime is relayed using Formula 4.3. Also, an estimation of the time it takes a packet s_i to reach a subscriber s is calculated as depicted in Formula 4.4.

$$deadline'_i = deadline_i - (t_p + t_s) \tag{4.3}$$

$$delay_i = t_s * D(s_i, s) \tag{4.4}$$

t_p expresses the time a packet is being processed at a node, namely from the moment it is received to the moment it is relayed. t_s refers to the average time that it takes to send a data packet to a neighbor node. Because t_s time cannot be measured in real-time unless the two nodes involved in the communication have synchronized their clocks, a constant

t_s value has been used. Whenever a deadline field of a packet is updated and the resulting value is less or equal to 0 the packet is discarded. Also if the $delay_i$ value calculated is higher than the remaining deadline the packet is discarded. In addition packets with the same priority are queued ordered by remaining deadline. This way, the smaller the deadline is the sooner they are handled compared with other packets in the same priority level.

4.3 Evaluation

In order to test the protocols described in this chapter, several different tests have been carried out. The tests are depicted in this section, with the analysis of the results obtained.

4.3.1 Environment set-up

The prototype of the algorithm has been implemented in C programming language for the TelosB motes running the Contiki operating system [110]. The resulting code has been simulated using the Cooja simulator [111]. The Cooja simulator emulates TelosB motes at machine code instruction set level. The communication model takes into account packet loss when nodes are transmitting at the same time.

4.3.2 Energy consumption, reliability and packet delay

The test scenario is composed of a network of 110 nodes arranged in a grid as shown in Figure 4.9. To test the routing algorithm, a set of nodes will periodically report a series of messages.

The test scenario has a total of 5 subscribers and 9 publishers dealing with data from 3 different topics: temperature (TM), humidity (HM), linear displacement (LD). Publishers/subscribers are marked with square/diamond/triangle symbols indicating that they produce/receive data from TM/HM/LD topics respectively. Node 51, represented as a star, is subscribed to all three topics and will therefore receive information from all 9 publishers. The communication in this scenario follows a many-to-many communication pattern that allows the features of our algorithm to be analyzed. Each node except for the ones in the boundaries has 8 neighbors at 1 hop distance. The test is made up of a series of data packets published at the same time from each of the publishers every 10 seconds. The total number

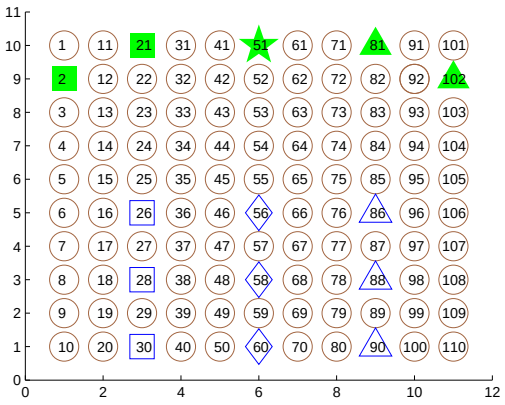


Figure 4.9: Evaluation scenario: 110 sensor nodes arranged in a grid. Square: HM topic. Diamond: TM topic. Triangle: LD topic. Circle: neither subscriber nor publisher. Star: all three topics. Filled/non-filled figures indicate subscribers/publishers

of packets sent by each publisher is 201. For example, information generated by node 26, 28, 30 will be received by subscribers 2, 21 and 51. Node 60 is configured to use a 3-retransmission reliability scheme. The rest of the nodes operate on a best effort basis. This scenario is used to study the reliability (delivery ratio), energy consumption (expressed in millijoules) and mean packet (expressed in ms) delay of each node. Before carrying out the test, the network has been initialized to let each node find all its neighbours.

To test the algorithm the following weight function has been used

$$
\begin{aligned}
f_f(s_j) &= \begin{cases} 0 & \text{if } E(s_j) < THRESHOLD \\ 1 & \text{otherwise} \end{cases} \\
f_w(neighbor\_status) &= E_m(s_j, neighbor\_status) \\
W_i(s_j) &= f_f(s_j) * f_w(neighbor\_status)
\end{aligned}
\tag{4.5}
$$

where neighbor_status contains all the information about the nodes in P_i. This weight function gives preference to neighbors with high remaining energy and will therefore help to balance the energy consumption between the possible relay nodes in the path between publishers and subscribers.

Figures 4.10 and 4.11 show the energy consumption for the test using/without using the PS-QUASAR multicast mechanism respectively. A series of colored dots have been used

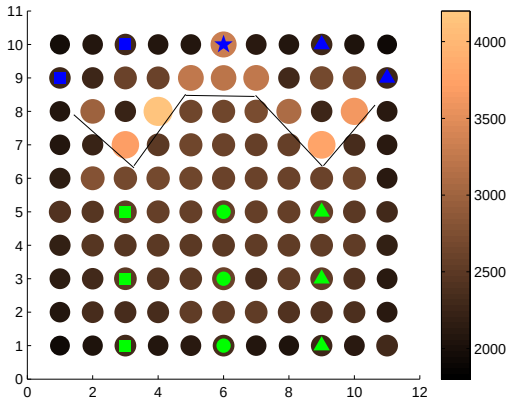


Figure 4.10: Energy consumption of the test scenario using PS-QUASAR multicast (energy is expressed in millijoules).

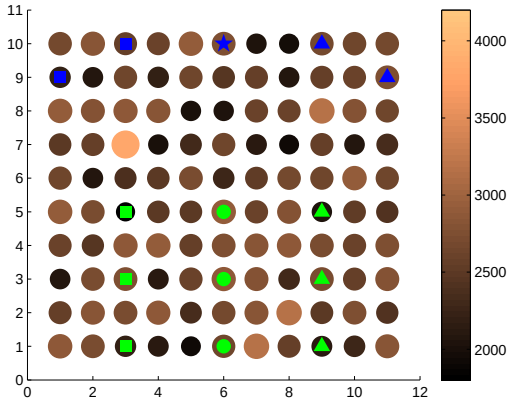


Figure 4.11: Energy consumption of the test scenario without using PS-QUASAR multicast (energy is expressed in millijoules).

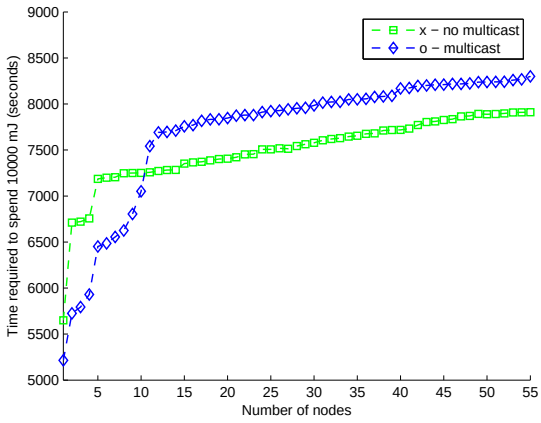


Figure 4.12: Time it takes a certain number of nodes in the network to expend 10000mJ of energy.

to represent the energy consumption. Each dot represents a sensor node in the network. The color of each dot represents the energy relative to the other nodes in the network, i.e. the darkest color corresponds to the node with the highest remaining energy after the test. When multicast is not used then a packet duplicate must be created everytime a message is sent even if multiple subscribers are in the same direction. Because each of these packets can take a different path towards the subscriber the energy consumption is less balanced as shown in Figure 4.11. On the other hand using the PS-QUASAR multicast mechanism optimal paths are detected in a fully distributed way and they are used to transmit less packets which translates to an improvement in energy consumption and reliability. This has been marked as a set of lines in Figure 4.10. Using this path all corresponding subscribers can be reached at minimum distance whilst avoiding duplicating packets. The mean energy consumption of the maintenance protocol without taking into account the data packets sent is of 369,32 milliJoules.

Comparing Figures 4.10 and 4.11 it can be seen that the energy consumption in Figure 4.11 affects a higher number nodes and is less balanced in the sense that nodes with a higher energy consumption are dispersed across the whole network. On the other hand, in Figure 4.10 only nodes within optimal paths show a relatively higher energy consumption. Optimal paths, in this context, refers to the paths at minimum distance that can be used by publishers to send the information to multiple subscribers at the same time.

The mean energy consumption during the test is 2474/2567 milliJoules for the scenario

Publishers		Subscribers: 2	21	51 (All)	81 (LD)	102 (LD)
TM	30	95,52	94,03	88,56	x	x
TM	28	99,50	99,00	95,52	x	x
TM	26	99,50	99,00	97,51	x	x
LD	90	x	x	91,04	95,02	99,50
LD	88	x	x	98,01	99,50	99,50
LD	86	x	x	95,52	99,50	95,52
HM	60	x	x	100,00	x	x
HM	58	x	x	98,01	x	x
HM	56	x	x	98,01	x	x

Figure 4.13: Delivery ratio of packets using multicast (in hundreds per cent)

Publishers		Subscribers: 2	21	51 (All)	81 (LD)	102 (LD)
TM	30	52,74	93,53	80,10	x	x
TM	28	68,66	94,53	93,03	x	x
TM	26	71,64	95,52	92,04	x	x
LD	90	x	x	46,27	94,53	87,06
LD	88	x	x	59,20	97,01	94,53
LD	86	x	x	59,20	99,50	94,53
HM	60	x	x	100,00	x	x
HM	58	x	x	97,51	x	x
HM	56	x	x	99,50	x	x

Figure 4.14: Delivery ratio of packets without using multicast (in hundreds per cent)

with/without multicast respectively. The result indicates that the mean energy consumption is lower in the scenario that uses multicast. In order to better illustrate the energy consumption in both scenarios Figure 4.12 shows the time it takes 50% of the nodes (55 nodes) in the network to spend a predefined value of 10000 milliJoules. Initially, the multicast scenario seems to spend the energy more quickly but after the first 10 nodes (which correspond to nodes near the subscribers) the nodes in the non-multicast scenario start spending the energy at a higher pace. The 3 graphs previously presented, however, do not explain the higher energy consumption in nodes near the subscribers that the multicast scenario has. In order to understand that, reliability ratios need to be taken into account.

Figures 4.13 and 4.14 show the delivery ratio in hundreds per cent for both cases. It is clear that all packets sent from node 60 (which is configured to sent data in a reliable way) have been received by node 51 in both tests. Also, a much higher delivery ratio is achieved using multicast. For example, without using multicast more than half of the packets between nodes 90 and 51 are lost. If a packet is lost it does not reach the subscriber and therefore does not generate an energy consumption in nodes near the subscriber. This is one of the reasons the energy consumption in nodes near the subscriber is higher in the case where multicast is used. In the test a total number of 4221 packets should be received if a 100% reliability is assumed. For the case where multicast has been used, 4095 have

Publishers		Subscribers: 2	21	51	81	102
		TM		Sink	LD	
TM	30	1303,7	1438,6	1605,3	x	x
	28	1035,0	1239,4	1322,4	x	x
	26	829,2	915,6	980,3	x	x
LD	90	x	x	1685,6	1462,8	1294,5
	88	x	x	920,4	1068,7	834,8
	86	x	x	642,8	885,4	709,8
HM	60	x	x	1593,3	x	x
	58	x	x	1235,1	x	x
	56	x	x	979,8	x	x

Figure 4.15: Mean delay of packets using multicast

Publishers		Subscribers: 2	21	51	81	102
		TM		Sink	LD	
TM	30	1095,4	1264,0	1457,1	x	x
	28	720,5	920,5	1029,5	x	x
	26	654,9	842,9	1010,8	x	x
LD	90	x	x	1546,0	1080,6	988,2
	88	x	x	1100,7	780,2	673,7
	86	x	x	678,7	576,9	482,1
HM	60	x	x	1402,1	x	x
	58	x	x	971,3	x	x
	56	x	x	926,4	x	x

Figure 4.16: Mean delay of packets without using multicast

been received. However, for the case where multicast is disabled a total of 3559 packets have been received. Even taking into account this difference in the delivery ratio, the mean energy consumption is lower in the multicast test.

The difference in the delivery ratio is produced by the collisions when transmitting at the same time. Because the multicast mechanism reduces the network traffic it lowers the probability of collisions which affects the delivery ratio. The results show that the multicast mechanism reduces the network traffic, improves the reliability and helps to save energy although the energy spent is more localized. The higher the number of subscribers in the same direction from the publisher, the greater the effect it has on the number of packets generated.

Publishers		Subscribers: 2	21	51	81	102
		TM		Sink	LD	
TM	30	-208,3	-174,6	-148,2	x	x
	28	-314,6	-318,9	-292,9	x	x
	26	-174,3	-72,7	30,6	x	x
LD	90	x	x	-139,6	-382,1	-306,3
	88	x	x	180,3	-288,5	-161,0
	86	x	x	35,9	-308,4	-227,7
HM	60	x	x	-191,3	x	x
	58	x	x	-263,8	x	x
	56	x	x	-53,4	x	x

Figure 4.17: Difference between mean delay of packets with and without using multicast

Publishers		Subscribers: 2 (TM)	21 (TM)	51 (Sink)	81 (LD)	102 (LD)
TM	30	100,00	96,08	100,00	x	x
	28	98,04	96,08	98,04	x	x
	26	100,00	100,00	100,00	x	x
LD	90	x	x	90,20	90,20	90,20
	88	x	x	100,00	100,00	100,00
	86	x	x	100,00	100,00	100,00
HM	60	x	x	96,08	x	x
	58	x	x	100,00	x	x
	56	x	x	100,00	x	x

Figure 4.18: Delivery ratio using a 3-retransmission reliability scheme and multicast

Finally, Figures 4.15 and 4.16 show the main delay packet with and without using multicast respectively. In addition Figure 4.17 shows the difference in delay between both cases. A negative value indicates that delay using multicast is lower. It can be seen that the multicast can deliver the packets with a lower delay. There are some cases where delay is actually higher than the test without using multicast but as explained before, this is due to the low delivery ratio obtained when multicast is not used.

4.3.3 QoS

In this section the QoS mechanisms provided by PS-QUASAR are studied. More specifically, the reliability, deadline and priority support of the middleware is analyzed by means of several tests.

4.3.3.1 Reliability

Reliability is achieved in our middleware by means of retransmissions. This has a direct effect on energy consumption, that is, the more reliable a retransmission is the more energy consumption it will produce in the network. We have seen that the multicast mechanism helps to save energy consumption by not duplicating packets unless strictly necessary. In this test we have sent 51 messages every 20 seconds with the same publisher/subscriber configuration as Section 4.3.2 but this time configuring all publishers with a 3-retransmission reliability scheme.

The delivery ratio of both scenarios is shown in Figures 4.18 and 4.19. As expected the delivery ratio in this scenario is much higher than the one obtained in the previous test (Figures 4.15 and 4.16). In the example where multicast is used the delivery ratio obtained is much better, reaching almost 100% in all subscribers. However, even using a

Publishers		Subscribers: 2	21	51	81	102
		TM		Sink	LD	
TM	30	98,04	86,27	76,47	x	x
	28	100,00	90,20	88,24	x	x
	26	86,27	98,04	82,35	x	x
LD	90	x	x	100,00	74,51	72,55
	88	x	x	98,04	92,16	72,55
	86	x	x	98,04	82,35	98,04
HM	60	x	x	94,12	x	x
	58	x	x	100,00	x	x
	56	x	x	96,08	x	x

Figure 4.19: Delivery ratio using a 3-retransmission reliability scheme and without using multicast

Node	Priority	Deadline (ms)
61	6	9000
43	6	10000
25	5	9000
7	5	10000

Table 4.1: Node QoS policy for the priority/deadline test

3-retransmission scheme the delivery ratio of the scenario where no multicast is used is significantly affected by the distance between publisher and subscriber. From the results obtained in both scenarios it can be seen that the network traffic is the main reason for the network performance. In the second scenario, the overhead produced by not using multicast has generated much more traffic that favors packet collisions and therefore has a direct effect on the delivery ratio. The mean energy consumption of the nodes in the scenario that uses multicast is of 1222,30 mJ where as in the scenario without multicast the consumption is of 1351,93 mJ. That supposes around 10% in energy savings by using the multicast mechanism. Because in the multicast scenario, packets generated are not duplicated unless strictly necessary they do not produce retransmissions that directly influence the energy consumption.

4.3.3.2 Deadline and priority

In this test the deadline and priority mechanisms explained in Section 4.2.4 are tested. Figure 4.20 shows the test scenario. Nodes 7, 25, 43 and 61 periodically publish information at the same time that it is received by node 1. This test simulates the burst of messages that is produced by a sensor network when an abnormal situation is detected in the environment.

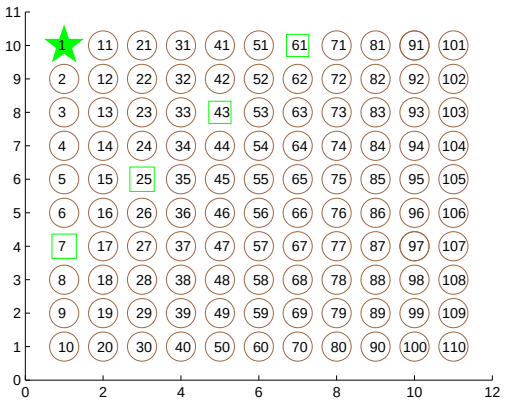


Figure 4.20: Test scenario for the deadline/priority test. Filled/non-filled figures indicate subscribers/publishers.

At that moment the network starts generating a high number of packets and they are all relayed at the same time to the subscribers. To simulate this scenario, one packet is published at the same time by each of the four publishers every 250 milliseconds. The total number of packets published by each node is 400. The QoS policy of each of these nodes is shown in Table 4.1.

Figure 4.21 shows the mean delay of the packets from the moment they are sent to the moment they are received in the subscriber. It can be seen that the packets with smaller delay are the ones with highest priority as expected. Also, packets with the lowest deadlines have a lower delay within a same priority value.

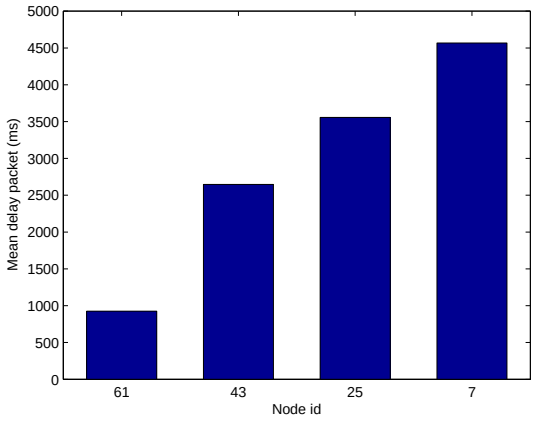


Figure 4.21: Mean packet delay in a scenario with nodes with different deadline and priority configurations (see Table 4.1)

5

Matching data link and network communication layers

This chapter analyzes and discusses the difficulty of providing a middleware supporting QoS, focusing on the influence of network traffic on network performance and the problem that common mechanisms for providing priority have in terms of delay reduction. More specifically, the chapter tests different network topologies and sizes by means of the PS-QUASAR middleware under different network loads and analyzes the results in terms of reliability, collisions, queue size and delay.

In conventional sensor devices, the unicast communication scheme used by developers and provided at the network layer conceals the shared medium that exists in lower layers and sometimes it is difficult to explain certain performance results if we do not take into account the way the data link layer works. Considering these issues and the need to provide a data link layer that is compatible with the features of the network layer, the performance of queue-based priority mechanisms is discussed, based on the counterintuitive results that

have been obtained. Section 5.1 describes the motivation. The case study scenarios are explained and analyzed in Section 5.2. The evaluation is given in Section 5.3.

5.1 Motivation

As previously stated, supporting real-time QoS and programming applications for WSANs is a challenging task. We have dealt with some of these challenges in our papers [64] [112] which deal with the CIP problem and we have personally experienced the issues identified in this chapter. Principally, two (not exclusive) approaches have been proposed to deal with priority and reliability. The first approach is to deal with these issues at the network layer. This means that the routing protocol is the one in charge of dealing with reliability and priority. In the other approach, these requirements are handled at the data link layer, i.e., the MAC protocol is in charge of these tasks.

The most common way of offering reliability is by means of multiple retransmissions of a packet until an ACK is received by the sender. As for priority, a PQS constitutes the most common mechanism. Packets with a higher priority are placed in the queue in the top positions so that they are handled more urgently. The use of reliability at the network layer is reasonable, as this layer is the one over which developers program their applications. Developers can control when to use the extra resources (energy and computational resources consumed by the retransmissions) necessary to provide a higher level of reliability. On the other hand, with regard to priority we have identified the inadequacy of dealing with priority only at the network layer. This means that, unlike reliability, a middleware supporting priority (by means of a PQS) should not be used on top of a data link layer that is not able to handle it.

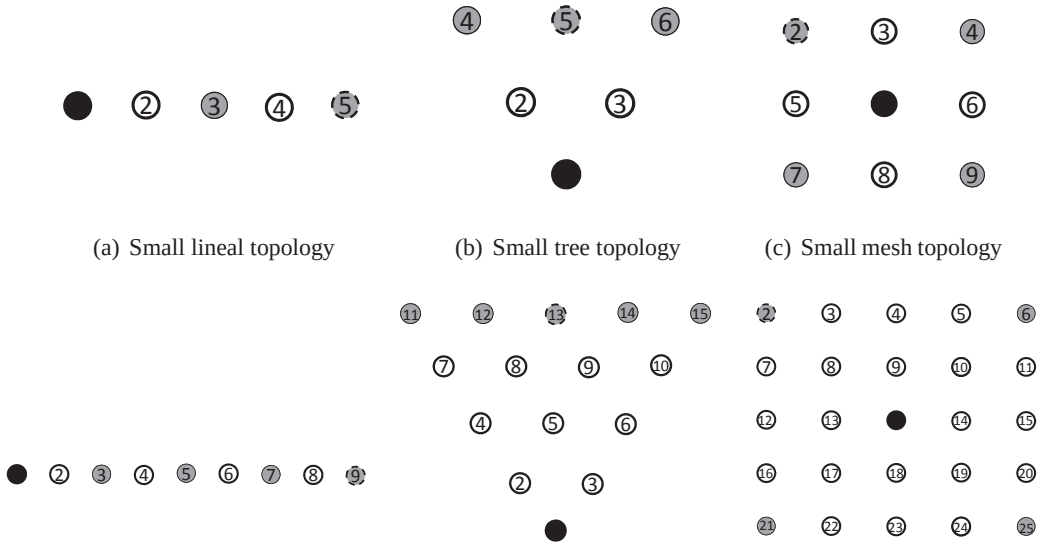
Ideally, priority should be handled at both layers. Each layer uses this information for different purposes but both cooperate to improve the performance of the communication protocol. For example, priority information at the network layer can be used to select optimal paths such as shortest paths or paths where a higher RSSI can be found to ensure that important packets arrive at their destination. On the other hand, the data link layer protocol is typically in charge of buffering packets and dispatching the important ones first. Also, assuming that a CSMA-based protocol is used as the MAC protocol, in circumstances where multiple nodes want to send, the MAC protocol should give the medium to the node

requesting to send the most important packet.

Another situation to bear in mind is when there are nodes which not only carry out sensing tasks but also have to act in order to control the environment (actor nodes). Depending on the application, there can be nodes that make their own decisions (local decision scheme) based on the information sent by their neighbour nodes or sensed by themselves. However, there are situations where it is advisable to study and analyze all the information sensed by the whole sensor network. To do this the sensor network sends all the information to a "control centre" (SCADA). Once the information has been processed and analyzed the decisions are sent back to the actor nodes. This communication scheme (from sensor nodes to the control center and from the control centre to the actor nodes) is more complex as traffic flows in different directions. These kinds of scenarios need to be studied carefully and are not covered in this chapter.

Regarding the goal of this study, the first tests we ran to asses the performance of PSQUASAR (provides a PQS at the network layer) were carried out over the ContikiMAC [113] data link layer protocol which does not handle priority. After carrying out some experiments to study the performance of the priority packets within the WSAN, the results showed that the latency and reliability of these kinds of packets were not much better than the performance of the non-priority packets, actually in some simulations they were even worse. In order to understand what causes this behavior, the performance of WSANs under different workloads and the effectiveness of PQSs, a deeper study of different scenarios is presented in the following sections, focusing on answering the following questions:

- How does the network behave as the network traffic increases?
- Are WSANs intrinsically scalable in size and throughput?
- To what extent does the network topology influence the WSAN's performance?
- Is a PQS an effective mechanism to deal with priority?
- In what circumstances does a PQS behave as expected?



(a) Small lineal topology (b) Small tree topology (c) Small mesh topology

(d) Big lineal topology (e) Big tree topology (f) Big mesh topology

Figure 5.1: Network topologies. Black: sink node. Grey: source node. White: router. (For test in Section 5.2.3. Dashed border: sends priority packets)

5.2 Case study

In this section the different scenario settings used in the test are described and their use is justified.

5.2.1 Network Topologies

In order to answer the questions presented in Section 5.1 a set of tests have been carried out using different topologies, namely mesh, tree and lineal topologies. These three kinds of topologies are the most commonly used to deploy WSANs.

In mesh networks all neighbor nodes are interconnected, forming a net structure. This allows nodes to use different paths towards their destination and introduces redundancy into the network which can be used to provide a high fault tolerance. However the structure of these kinds of networks make them more prone to collision and interference between nodes as nodes usually have multiple neighbors. Examples of applications where mesh

networks are used are forest fire detection [114] or vehicle tracking [115]. Tree networks can be considered a subcategory of mesh networks but with some special features. In tree networks there is usually a single sink node or base station. Multiple nodes report to the sink node by sending the packets through the tree topology previously discovered by a discovery protocol. Each node is then able to send messages to the sink node by relaying the message to the parent node according to the tree topology. For example, a hierarchical tree-based network topology is used for landslide detection [116] or traffic monitoring [117].

Lineal networks, on the other hand, constitute an important topology in CIP systems. Examples of applications using these lineal networks are two demonstrators deployed in the context of the European project WSAN4CIP [64] in which we participated. The first one was a monitoring system for an electricity distribution infrastructure and the second a monitoring system for a water distribution system. Due to the intrinsic nature of both systems it is the lineal topology that best matches the topology of these infrastructures. Figure 5.1 shows the network topologies used in all the tests. Two different sizes have been used for each network topology to better understand how a change in size affects the results. Each network has a single sink node (marked in black) which receives information from multiple source nodes (marked in grey). All packets in the tests are filled up to the maximum capacity (approximately 108 bytes) and sent in unicast with no retransmissions. These experiments are organized into three different groups explained in the following sections.

5.2.2 Sampling Rate Vs Network Performance

A PQS only makes sense when there is at least a minimum of network traffic, in other words there are a sufficient number of packets in the network so that queues need to be used to cache packets until they can be processed. For this reason, the first set of experiments aims to provide an insight into how the network behaves as the network traffic increases. More specifically it studies the relationship between sampling rate and network saturation. On the one hand, we study what the relationship is between the sampling rate of the nodes (network traffic) and parameters such as MAC queue occupation rate, end-to-end delay and end-to-end reliability, and on the other hand, it allows us to discover the proper sampling rate frequency that generates enough traffic to saturate the network. It also gives an indication of how the different network topologies deal with high traffic. These tests are necessary as a PQS is highly dependant on network traffic.

5.2.3 Network Layer Vs Data Link Layer

The goal of these experiments is to analyze the effectiveness of a PQS. In the communication protocol stack there are two layers where this system can be implemented: the MAC layer and/or the network layer. We try to identify the best place to implement the PQS and the relationship between network layer capabilities and MAC layer capabilities. More specifically, four different setups are compared (in terms of reliability and delay): (1) WSAN without PQS, (2) WSAN with a PQS at network level, (3) WSAN with a PQS at MAC level and (4) priority at network and MAC level. In this test, only the packets from one node for each network topology are configured to be priority. The rest of packets in the network are non-priority ones. The node which sends the priority packet in this particular test is marked with a dashed border in Figure 5.1. The expected behavior of an optimal PQS is that packets from these priority nodes are given priority over packets from the rest of the source nodes in terms of a lower delay and higher reliability. The intuitive behavior expected is to achieve a lower delay at the expense of delaying non-priority packets.

5.2.4 Priority Queueing System Performance

In this test, the performance of the PQS is studied, based on the number of priority packets there are in the network. To do so, each source node is programmed to send priority packets with a given probability. This test introduces unpredictability in the reporting pattern of priority and non-priority packets and gives us a way to compare the behavior of the network with a different proportion of priority packets in it at a time. It emulates a real scenario where the time events occur, cannot be predicted.

These experiments allow us to analyze: (1) whether a PQS helps important packets to increase their reliability and decrease their delay and (2) to what extent this is possible based on the proportion of priority packets in the network.

In Section 5.3, all the results obtained from the aforementioned experiments are analyzed in detail.

5.3 Evaluation

This section presents the evaluation tests and their results. All the tests have been repeated for each network topology presented in Figure 5.1. However, for the sake of clarity, in some cases figures are limited to certain topologies (the most representative ones). All figures and results obtained can be accessed at the following address http://pqs.lcc.uma.es.

5.3.1 Environment set-up and scenario settings

The application scenario was implemented in the C programming language for the Tmote-sky motes running the Contiki operating system [118]. The resulting code was simulated using the Cooja simulator [119]. The Cooja simulator emulates Tmote-sky motes at machine code instruction set level. The communication model takes into account packet loss when nodes are transmitting at the same time, i.e., collisions were taken into account in the simulation.

5.3.2 Network and Data Link layer setup

As detailed in previous sections PS-QUASAR and ContikiMAC have been used as the protocols for the network and data link layer, respectively. PS-QUASAR allows the network to self-organize in order to deliver information from publishers to subscribers. For example in the network topologies shown in Figure 5.1, each source node is configured to be a publisher and the sink node is the only subscriber. Once the network has been deployed, nodes automatically coordinate and exchange information to learn the path towards the subscribers. PS-QUASAR has a built-in PQS to handle the priority packets (at the network layer). PS-QUASAR does not check for an idle state of the radio before sending the packet to the data link layer. It simply analyzes packets and in the case it has to retransmit them, sends them to the MAC layer.

The MAC protocol used in Contiki OS is called ContikiMAC. It is a CSMA protocol that makes use of radio duty cycling (RDC) to power off radios whenever possible in order to save energy. ContikiMAC comprises two layers: the RDC layer and the CSMA layer, the higher one being CSMA. By default, packets from the network layer sent to the ContikiMAC are placed in a FIFO queue at the CSMA layer, and processed by order of arrival. This means that ContikiMAC does not handle different traffic classes. Access to the medium is

Setup	PS-QUASAR	ContikiMAC
1	Handles priority	Does not handle priority
2	Priority disabled	Does not handle priority
3	Priority disabled	Handles priority (modified version)
4	Handles priority	Handles priority (modified version)

Table 5.1: Network and MAC protocol setup

obtained depending on the time each node requests to send and how busy the medium is (backoff mechanism).

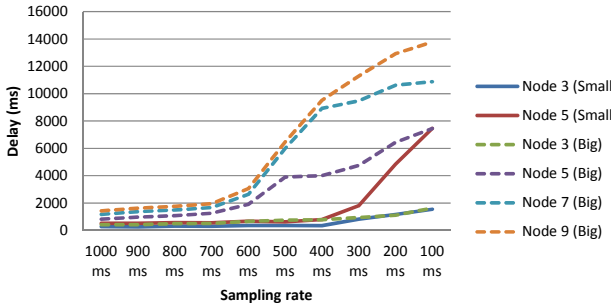
This setup by default (PS-QUASAR over ContikiMAC) only handles priority packets at the network layer. This means that priority information is not taken into account by the data link layer which inserts packets into the FIFO queue as they arrive. In order to test different configurations we have modified the ContikiMAC implementation to handle priority. Priority is expressed as a number, the higher the number, the higher the priority given to a packet. In this new priority version of ContikiMAC, packets are now enqueued by priority rather than by order of arrival.

The four different setups used in the tests are depicted in Table 5.1. Setups 1, 2, 3 and 4 are compared in Section 5.3.4.

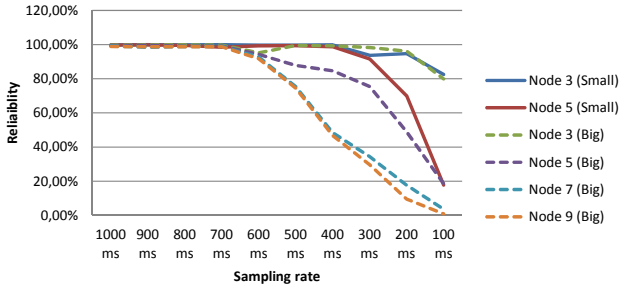
5.3.3 Network performance

Figure 5.2 shows the results obtained from the experiments described in Section 5.2.2. For a particular sampling rate, 1000 messages have been sent from each source node to the sink node. This experiment has been repeated 10 times for each sampling rate and mean values have been used. Observing all the graphs as a group, it can be appreciated that the general performance of the network starts to decrease when the sampling rate frequency passes a threshold that varies for each topology.

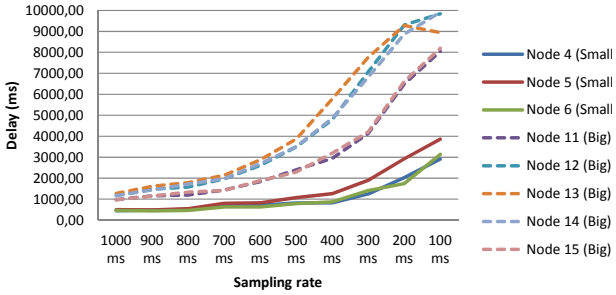
Table 5.2 shows the different sampling rates and the associated throughput of each topology when congestion starts to appear. We note that the number of publishers in each scenario varies, as well as the distance between source nodes and sink nodes and therefore it is difficult to compare the results. What it is clear, is that the throughput of sensor networks



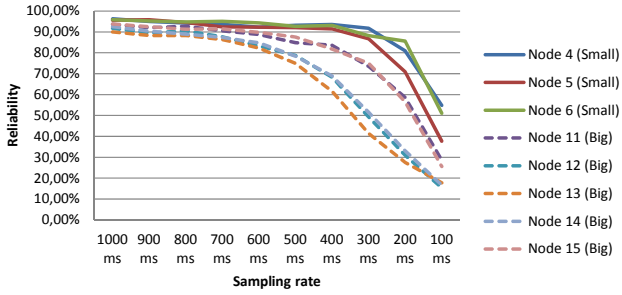
(a) Lineal topology delay results



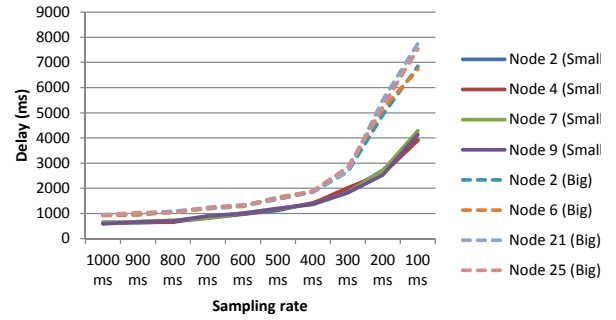
(b) Lineal topology reliability results



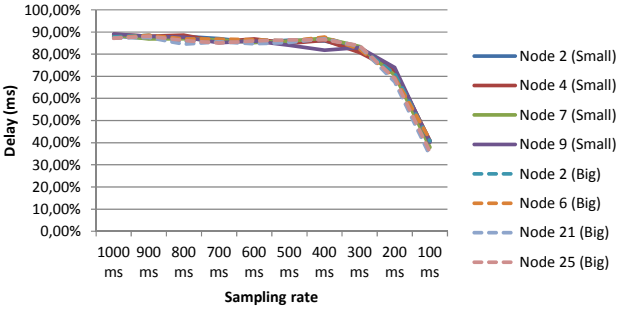
(c) Tree topology delay results



(d) Tree topology reliability results



(e) Mesh network delay results



(f) Mesh network reliability results

Figure 5.2: Network topology delay and reliability

Topology	Sampling rate (ms)	Source nodes	Throughput (kbps)
Lineal small	300	2	5,76
Lineal big	600	4	5,76
Tree small	300	3	8,64
Tree big	600	5	7,3
Mesh small	400	4	8,64
Mesh big	400	4	8,64

Table 5.2: Throughput and sampling rate of different networks when congestion starts to appear

is really low compared to the theoretical maximum which is around 40 kbps.

Moreover, the distance between the source nodes and their sink greatly affects the reliability and the delay. In all topologies, nodes that are far from the sink obtain worse results in terms of reliability and delay. For example, in the big tree topology, nodes 11, 12, 13, 14 and 15, which are the ones farthest from the sink node, show the highest delay and worse reliability (Figures 5.2(c) and 5.2(d)). However, packets from nodes 11 and 15 achieve better results than those from 12, 13, 14. This is due to collisions and traffic. Nodes 11 and 15 only have one neighbor (nodes 7 and 10, respectively) whereas node 12, 13 and 14 have two. In the end the transmission is carried out in a shared medium and therefore the more nodes there are in a neighborhood the more likely it is that collisions and delays are produced as nodes have to coordinate with each other to share the medium. Collisions, packet discards and delays due to existing traffic in the neighborhood of each node constitutes, in our opinion, the most important reason why there is a lack of reliability in WSANs. The reason for a bottleneck in a WSAN generally lies in the shared medium. Although each node constitutes an independent device, the communication is carried out in the same medium. In ContikiMAC, for example, CSMA is used, which means that each node has to wait for the medium to be free before being able to transmit.

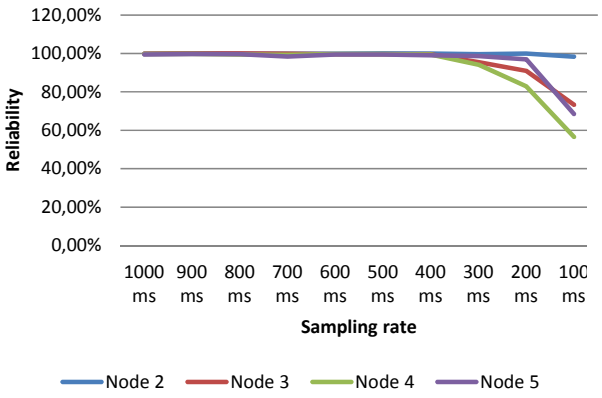
These findings may seem obvious but they can be applied to the WSAN design in the form of the following rule “the smaller the network the better its performance, the fewer neighbors each node has the better it will perform”. This means that in a real deployment it is advisable to apply clustering techniques to big networks to subdivide them into smaller

ones. In this way, the maximum distance between nodes decreases which has a direct influence on delivery ratio and delay improvement. For example, this is the approach taken in [112] where different clusters are formed, each working on a different radio channel to avoid interference. We therefore believe that current WSANs are not scalable in size, or in throughput (without using clustering techniques).

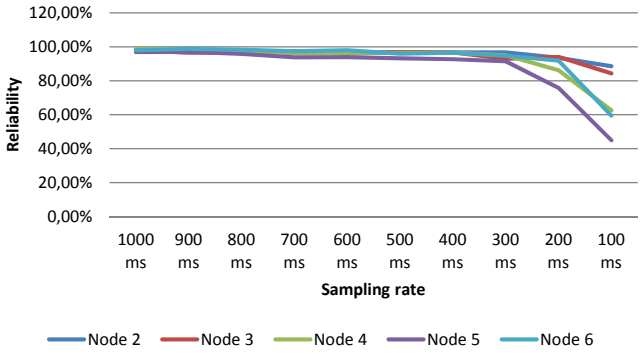
The influence of network traffic on the delay and delivery ratio is also greater on those nodes far from the sink. In Figures 5.2(a) and 5.2(b), for example, the delivery ratio and delay of nodes farthest from the sink decrease/increase, respectively, at a much higher rate than, for example, node 3 which is the nearest source node. This is mainly because the farther a node is, the more it has to be retransmitted but more importantly the more it has to compete for the shared medium. In congested networks, in particular, this constitutes a considerable amount of time which can cause packet discards.

Regarding the lack of reliability, Figure 5.3 investigates the origin of packet loss and its relation to collisions. It shows the percentage of packets that have been successfully sent at one-hop distance as well as the percentage of collisions and packet discards for the small lineal and small tree topology. The results agree with those shown in Figure 5.2. The number of collisions in the lineal network (Figure 5.3(e)) remains relatively small due to the low number of neighbors each node has. In the case of the tree topology the collisions are much more frequent since each node can have up to 4 neighbors. Collisions are present in all scenarios and do not seem to be greatly influenced by the sampling rate. Collisions depend on the time each node wants to access the medium at the same time as others. Since each node can be powered up at certain instants of time it is difficult to predict collision behavior. In any case, from the graphs it can be concluded that collisions are not the most important cause of the lack of reliability.

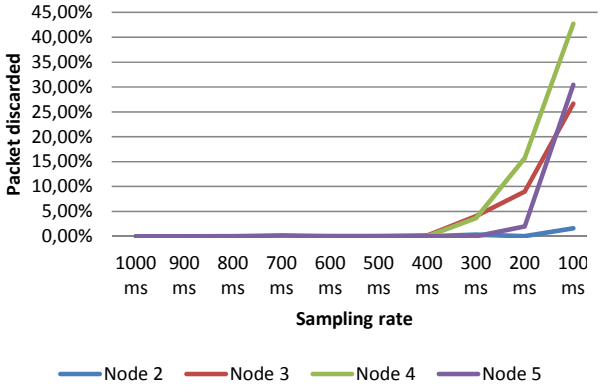
The main source of packet loss turns out to be the packet discards which are shown in Figures 5.3(c) and 5.3(d). Again, the shared medium cannot be used by more than one node in a neighborhood which causes the effective transmission rate to drop. As the sampling rate increases the rate at which messages are received is greater than the rate at which they are generated. This causes the MAC layer queue to fill up and start discarding packets. In addition, collisions and intense traffic have an effect of decreasing effective transmission rates which in turn favors packet discards.



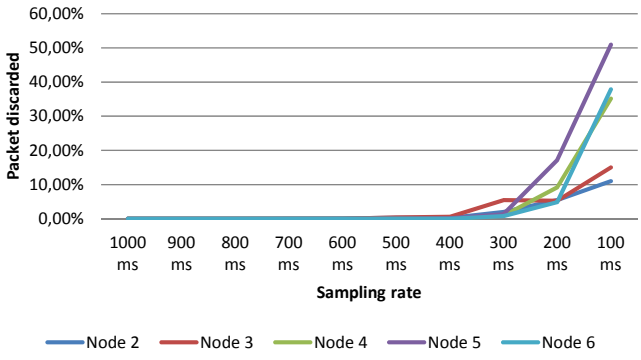
(a) Lineal topology one-hop reliability results



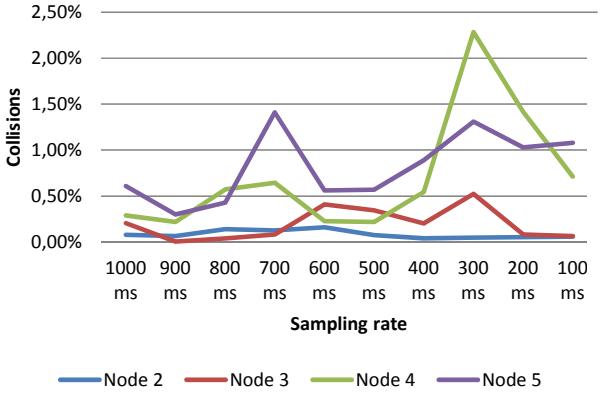
(b) Tree topology one-hop reliability results



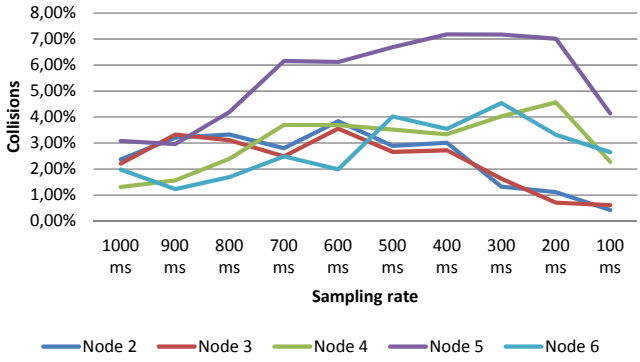
(c) Lineal topology one-hop discard results



(d) Tree topology one-hop discard results



(e) Lineal topology one-hop collision results



(f) Tree topology one-hop collision results

Figure 5.3: One hop results for lineal and tree topology

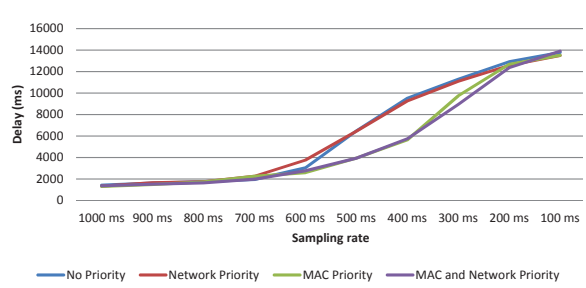
5.3.4 Network Layer Vs Data Link Layer

The goal of this test is to measure the performance of a PQS by comparing the reliability and end-to-end delay of priority packets in a WSAN with different topologies. For each of the topologies the four different stacks (referred to as setup 1, 2, 3 and 4 for the rest of the section) presented in Table 5.1, have been tested. In each test, each source node sends a total of 1000 packets to the sink node. Only one node in each topology is configured to send priority packets (marked with a dashed border in Figure 5.1). Each test is repeated 10 times for setup and for each sampling rate, resulting in a total of 300 individual tests for each topology. Delay and reliability mean values of each set of 10 experiments have been used.

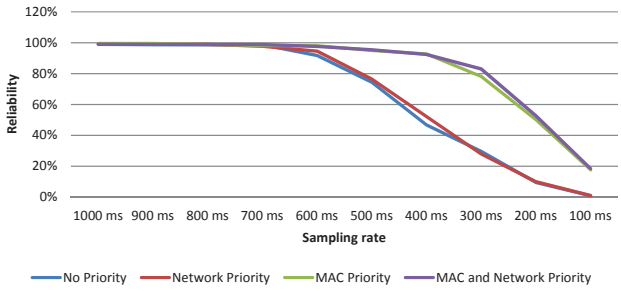
For the sake of clarity only the graphs for the big topologies are presented in this chapter. The rest can be found on the web page indicated at the beginning of Section 5.3. Figure 5.4 shows the results of the tests. These figures only show the delay and reliability of the priority node of each topology. For example, Figure 5.4(a) shows the mean delay of packets from node 9 for the topology depicted in Figure 5.1(d).

The first tests were carried out using setups 1 and 2. We expected the PQS at the network layer to perform significantly better than setup 1. The results obtained were however, quite disappointing, as there was no significant improvement in either reliability or delay when the PQS was used, actually it was almost the same (red and blue lines in Figure 5.4). This led us to analyze the MAC layer in more detail. MAC layers are in charge of sending packets in a one-hop manner. To do so, they coordinate with their neighbors so that only one node in a neighborhood is using the medium at any time, thereby avoiding collisions. However, if the medium is busy the MAC layer needs to cache packets until they can be sent. Therefore packet queues are essential at the MAC layer.

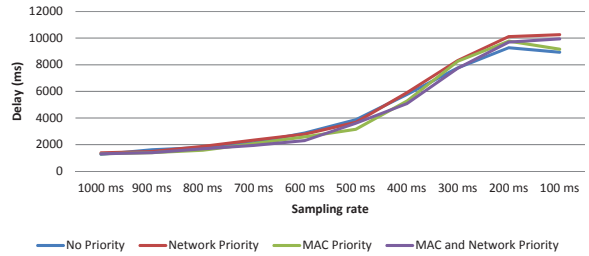
ContikiMAC, which is the default MAC protocol for the Contiki operating system, uses a FIFO that can hold up to 16 packets and does not deal with priority. Setup 2 uses a PQS at the network layer but the default ContikiMAC protocol at the MAC layer. This means that packets are ordered by priority at the network layer but in order of arrival at the MAC layer. Because the processing at the network layer is really fast (packets are not delayed at the network layer) they are sent to the MAC queue as soon as they arrive. This stops the packet queue at the network layer growing and therefore the PQS is not effective at this



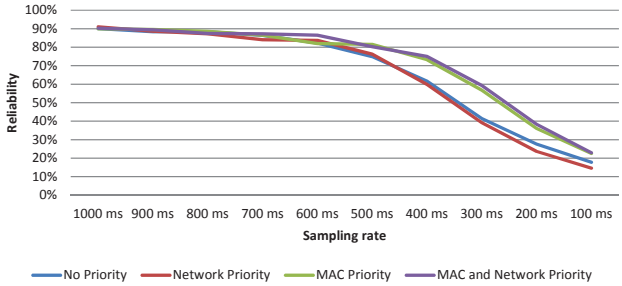
(a) Big lineal topology delay



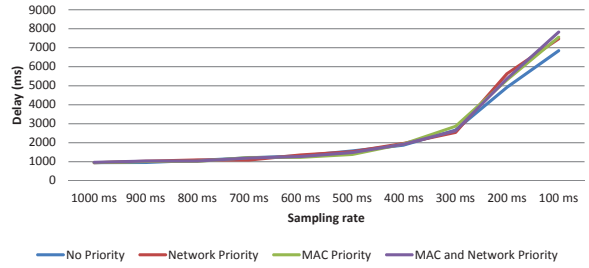
(b) Big lineal topology reliability



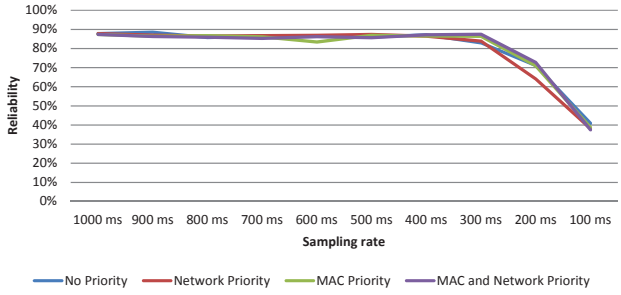
(c) Big tree topology delay



(d) Big tree topology reliability



(e) Big mesh topology delay



(f) Big mesh topology reliability

Figure 5.4: Delay and reliability of the priority node for big lineal, big tree and big mesh topologies with different setups

layer unless packet processing at this layer requires a significant amount of time. That is the reason why there is not a significant difference in terms of performance between setups 3 and 4.

One possible option to solve this problem would be to enqueue packets at the network level and only dispatch them to the data link layer when the MAC layer queue is empty. This option however, does not allow the queue in the data link layer to grow. In some specific MAC layers, optimizations are applied, to send packets in a burst, so if this option is implemented, the data link layer cannot apply these optimizations. An example of a data link layer with these optimizations is ContikiMAC which, where possible, sends multiple packets in a burst. Because packets on their way to the sink node are mostly delayed at the MAC layer a better option is to move the PQS from the network layer to the MAC layer.

The results depicted in Figure 5.4 show that the difference in delay and reliability for setup 1 and setup 2 are not significant. However in setup 3, which introduces PQS at the MAC layer, results are much better, mainly in terms of reliability. Three different conclusions can be drawn from these findings. The first one is that PQS at the MAC layer is effective in improving reliability and to some extent in reducing delay. The second one deals with the relationship between reliability and delay. The improvements in performance achieved with setup 3 are mainly in terms of reliability and much less in terms of delay. This means that there is an important tradeoff between reliability and delay which makes it difficult to simultaneously achieve a high reliability and a low delay. The third conclusion concerns the fact that the PQS behaves differently depending on the network topology. Note that PQSs are only effective when queueing occurs in the nodes. Therefore topologies which can balance the load of packets (for example by having different paths from source to the sink node) have a lower queue occupation ratio and therefore PQSs are less effective in those scenarios. Figures 5.4(b), 5.4(d) and 5.4(f) confirm this. Lineal topologies cannot balance the load because there is only one path from source node to destination and congestion is a common issue. In addition, this means that queues start to grow easily and a PQS is effective in these circumstances (see Figure 5.4(b)). Tree and mesh perform better in terms of load balancing and therefore the differences between setup 3 and the other setups is smaller for these two topologies. Proof of this is that Figure 5.4(f) shows that there is not a big difference between the different setups for the mesh topology. In our test topologies, the mesh topology performs better than the tree topology because source nodes are farther

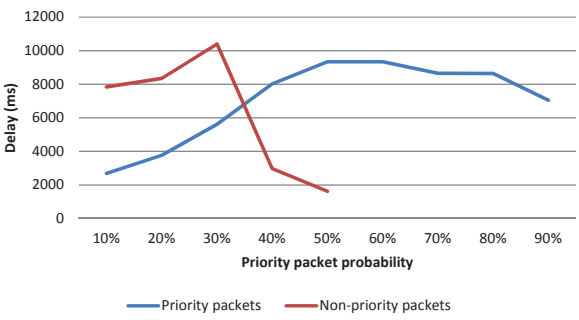
away from each other.

This first analysis encouraged us to carry out a deeper analysis on how the PQS behaves and what the relationship between the number of priority packets and reliability/delay is. The results obtained are explained in the next section.

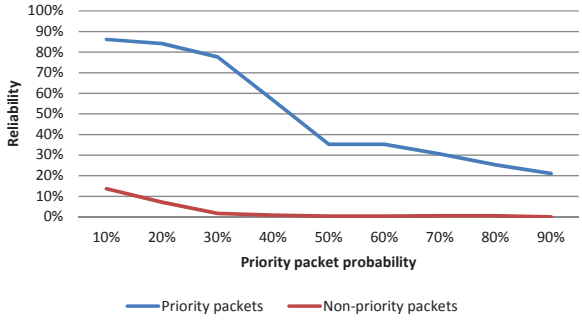
5.3.5 Performance of priority mechanisms based on queues

In the previous sections the need to match the MAC and network layer has been identified. A PQS at the MAC layer has been tested and it can be concluded that it effectively helps WSNs to give more importance to priority packets in terms of an improvement in reliability and to a lesser extent, delay reduction. In this last section a PQS at the MAC layer is studied to analyze its performance based on the number of priority packets in the network. The test is carried out for each network topology presented in Section 5.1. In each test all source nodes send a total of 1000 packets. Each packet has a defined probability of being a priority packet. This means that all source nodes in each network can send priority and non-priority packets. The experiment has been repeated with different values of probability ranging from 10% to 90% and delay and reliability results have been collected. Ten tests have been carried out for each priority packet probability and mean values are used. A 200 ms sampling rate has been used for all network topologies (except the small lineal topology) as this rate makes the PQS work and produces discarded packets, and therefore the end-to-end delay and reliability of the priority packets should improve with regard to the non-priority packets. This experiment allows us to analyze the network performance with different proportions of priority packets in the network and with an unknown reporting pattern.

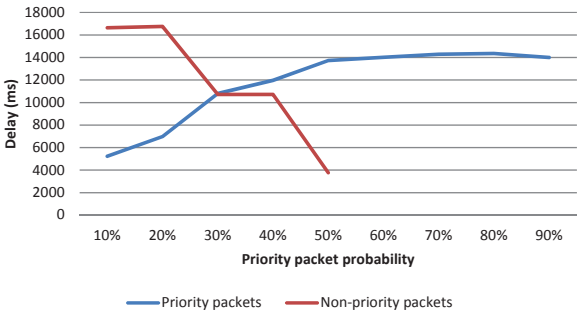
Figure 5.5 shows the mean delay and reliability for different network topologies with a different proportion of priority packets in the network for one source node (the rest of the nodes exhibit a similar behavior). In general, it can be seen that the delivery ratio is better and more stable for packets with a high priority which indicates that PQSs are indeed effective in handling high priority packets in networks with high traffic. However, there are significant differences in performance between the network topologies. Again, the smaller the network is (shorter paths between source and sink nodes) the better it performs. Other factors that have a direct influence on performance are the size of the neighborhoods (the smaller the better) and the number of source nodes in the network. Figure 5.5 shows that



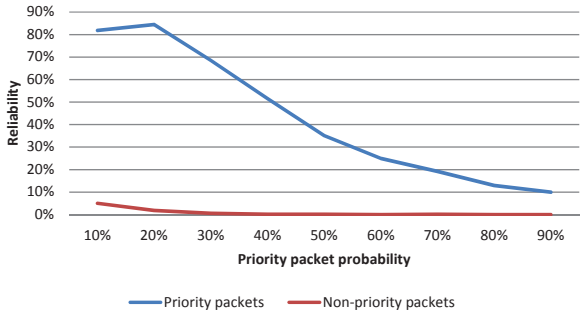
(a) Delay of small lineal topology (node 5, sampling rate 100ms)



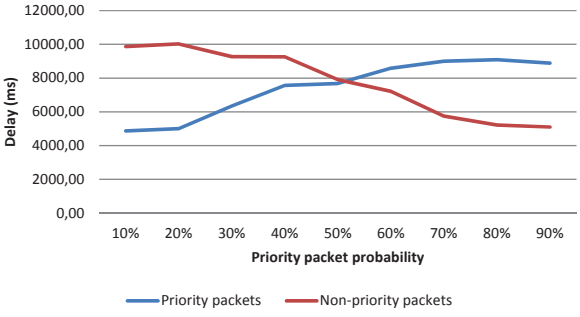
(b) Reliability of small lineal topology (node 5, sampling rate 100ms)



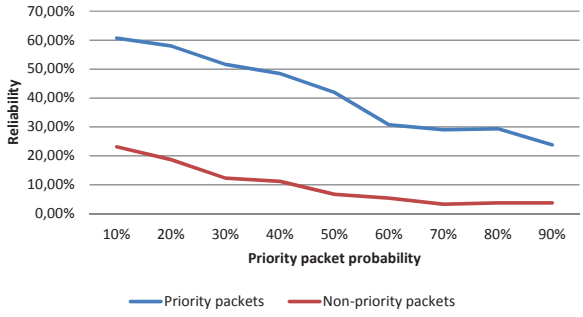
(c) Delay of big lineal topology (node 9, sampling rate 200ms)



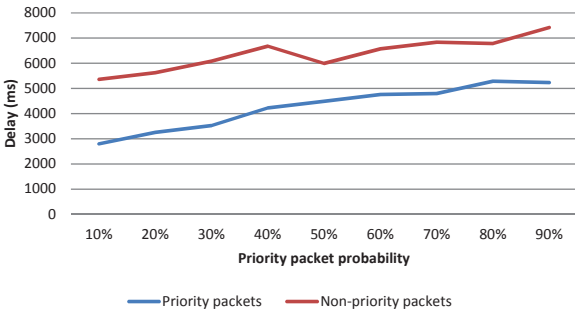
(d) Reliability of big lineal topology (node 9)



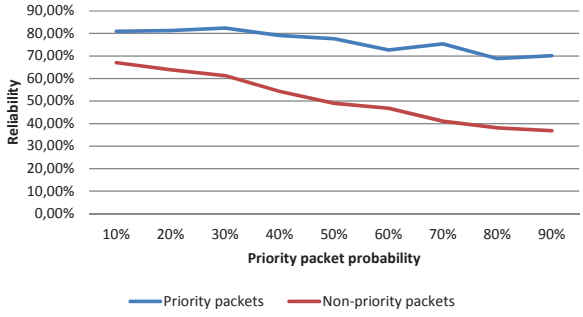
(e) Delay of big tree topology (node 13, sampling rate 200ms)



(f) Reliability of big tree topology (node 13, sampling rate 200ms)



(g) Delay of big mesh topology (node 2, sampling rate 200ms)



(h) Delay of small mesh topology (node 2, sampling rate 200ms)

Figure 5.5: Delay and delivery ratio depending on the number of priority packets sent

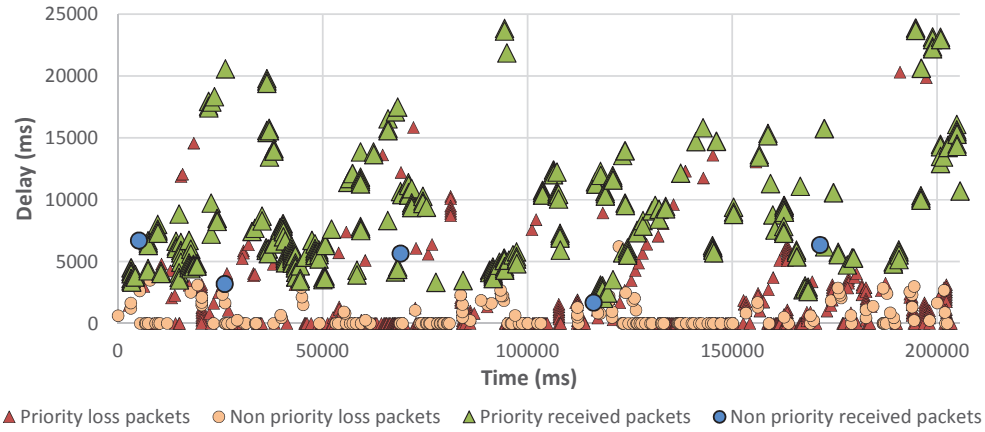


Figure 5.6: Packet reception time at sink node and packet delay for a single scenario (big tree topology) with an 80% priority packet probability

the better the network topology balances the load, the better it performs as the proportion of priority packets grows in the network. Mesh networks perform well because there are different paths between source and sink nodes that can be used by the routing protocol. Tree topologies also perform better than lineal networks but slightly worse than mesh networks. In our evaluation, tree topologies perform worse than mesh ones because source nodes are nearer to each other. These results relate to the ones presented in Figure 5.2 where performance of topologies such as a mesh or tree topology degrades more slowly compared to that of lineal topology.

On the other hand, contrary to popular belief, the mean delay is not strictly better in packets with high priority compared to non-priority ones. In fact as can be seen from Figures 5.5(a), 5.5(c), 5.5(e) there is a clear tendency for non-priority packets to reduce their delay and for priority packets to increase their delay as the proportion of priority packets in the network increases. Let us note that all the results shown in Figure 5.5 have been obtained using a 200 ms sampling rate except for the small lineal topology (Figures 5.5(a) and 5.5(b)) that uses a 100 ms sampling rate. The small lineal topology is really small and using a sampling rate of 200 ms does not saturate the network in a way that allows us to see this behavior. The same thing happens in mesh networks, a higher sampling rate is necessary for delay to show this behavior due to their load balancing capabilities.

In order to understand why the mean delay of non-priority packets decreases as the proportion of priority packets grows, single scenarios have been analyzed in order to study the

delay of individual packets as the simulation runs. Figure 5.6 shows the time each of the 1000 packets of a single test for the big tree topology is received, and its delay. Time 0 is given to the first packet that is received by the sink. Packets marked with a triangle correspond to priority packets whereas packets marked with circles correspond to non-priority packets. Lost packets are shown in red or orange. The delay of lost packets corresponds to the exact time they got lost on their way to the sink. It can be seen that a high proportion of priority packets are received whereas almost all non priority packets are discarded. High priority packets are given more importance and therefore they are delivered, although they have to wait in the queue for a significant amount of time. However, non priority packets are discarded if queues are full, in favor of priority ones. In the simulation overall, only 5 non-priority packets of approximately 200 have reached their destination. This happens in isolated cases where queues are momentarily not full and therefore delays are not high for those packets. The mean value gives the false impression that delay is actually equal or lower in non-priority packets than in priority ones when the ratio of priority packets in the network is high. When the network is highly congested, non-priority packets are discarded and their delay is therefore not used to calculate the mean value. To the contrary, priority messages are delivered, although with a higher delay as the network is congested.

In order to understand this behavior reliability needs to be taken into account. PQSs allows reliability to improve but contrary to what it is expected delay does not improve significantly. As mentioned before this is because a tradeoff exists between delay and reliability. In addition, if reliability is really low as in the case of non priority packets, mean values are unreliable as they are more likely to show a high variance.

To summarize the different conclusions drawn in this last section we can conclude that PQS at the MAC layer effectively improves the performance, mainly reliability, of priority packets. PQSs are effective when queueing occurs at the MAC layer and, as expected, the higher the proportion of priority packets in the network the worse the performance is. Lastly, in highly congested networks, PQSs constitute a simple yet effective way of providing priority, however mean results must be handled with caution when analyzing the effectiveness of this mechanism since they can sometimes be misleading.

6

Case study: Railway infrastructure health monitoring

In this chapter WSAN technology is applied to the CIP problem, more specifically to the monitoring of railway infrastructures. The application scenario consists of a railway bridge in which structural health is monitored. The WSAN is deployed along the bridge and takes periodical readings about the structural health of it. Trains passing through are used as data mules to get the information from the sensors which means that no direct connectivity to the internet is required for the WSAN. In order to tackle the lack of QoS support and the low level of abstraction of the sensor devices the PS-QUASAR middleware described in Chapter 4 has been used. The application proposed in this chapter tackles the use of WSANs in the CIP problem. Also, it proves and defends that the use of middleware abstraction such as PS-QUASAR can considerably simplify the task of developing WSAN applications and make it less error-prone. Finally, it makes use of interesting mechanisms that can be used to organize the network, such as clustering (to avoid packet collision and packet loss), data

fusion (to minimize the number of sent packets) and QoS support.

The rest of the chapter is organized as follows. Section 6.1 describes the motivation of the application scenario. In Section 6.2 related work is presented. The application architecture and implementation details are depicted in Section 6.3. The evaluation is carried out in Section 6.4.

6.1 Motivation

Railway infrastructures, as any other kind of infrastructure, are affected by the aging process. This is particularly important in this domain. For example, large sections of the railway lines in the United States were built in the late 19th century or beginning of the 20th century. In Europe large sections of the railway lines were reconstructed after the Second World War. Therefore, it is really important to regulate maintenance and restoration guidelines to ensure the safety in the railway transport. In this regard, more attention has been paid to this issue from the late 20th century onwards. The document containing the guidelines for the maintenance of the Spanish railway lines (ITPF-5) is regulated in the FOM/1951/2005 Ministerial Order [120]. In particular, for railway bridges, the guidelines establish that a visual inspection of elements of the infrastructure needs to be carried out every 15 years by specialized technicians. Furthermore, a general visual inspection is completed every year by non-specialized railway line guards.

This is sufficient for most railway bridges. In structures with unusual topology or particularly high/long structures, however, the information on the evolution of defects is more limited. Moreover the visual inspections are much more difficult to carry out and require a tempory closure to traffic. In these structures, it is common to check the state of the structure using specialized equipment or even install a permanent monitoring system, e.g. fiber optic instrumentation with BOTDA (Brillouin Optical Time Domain Analysis) or distributed sensor instrumentation. One of the main disadvantages of these systems is the high cost. Also, if there is no mobile coverage then data acquired by the system cannot be sent to the remote control center.

The current WSAN technology can be used as a permanent monitoring system and considerably reduce the cost of installation and maintenance since no wiring is required. The application presented in this chapter seeks to provide a system to monitor railway

infrastructures using WSANs cost-effectively. It also copes with the network coverage problem and tackles the transfer of large quantities of data in a reliable manner.

6.2 Related work

The use of WSANs for infrastructure health monitoring has been extensively studied. This section covers some of the existing proposals that focus on WSANs monitoring the infrastructure health of bridges.

In [121] a WSAN consisting of 20 sensor nodes is deployed on a road bridge to gather accelerometer and strain data. Nodes are assigned a sequential time offset based on their local addresses to enable them to transmit without collisions. Although TinyOS is used as the operating system, low level software is programmed to achieve higher data throughput. An actual deployment of a WSAN for railway bridge monitoring is described in [122]. The WSAN consists of 8 nodes that are deployed on the bridge and collect strain information whenever a train crosses the bridge. The network self-organizes as a routing tree to relay the information to a sink node. The information is then relayed from the sink node to the remote control centre using UMTS. In [123] a WSAN is used to monitor railway track status. Sensor devices are hierarchically organized with redundant paths. Multi-path routing is used to send the information to the remote base station. Fuzzy logic techniques are employed to aggregate data collected. In BriMon [124] a wireless sensor network composed of Tmote-sky devices is deployed on a railway bridge. Information is collected by nodes and retransmitted to the train that acts as a mobile sink node. The routing protocol forms a tree rooted at the head node of the WSAN by periodically transmitting a message which is flooded down the WSAN. The feasibility of the mobile data transfer from the WSAN to the train is studied by means of an experiment that only takes into account the mobile head node and the WSAN head node. Other real deployments of WSANs on road bridges are presented in [125] [126] [127] and [128].

Although, some of these approaches and the proposal covered in this chapter share some commonalities there are some important differences. Most of these proposals concentrate on the sensor processing part and a great number of them lack a general purpose routing protocol. BriMon, is the only one to take the data muling technique into account. Although it studies many different issues and aspects of the application by means of isolated testing

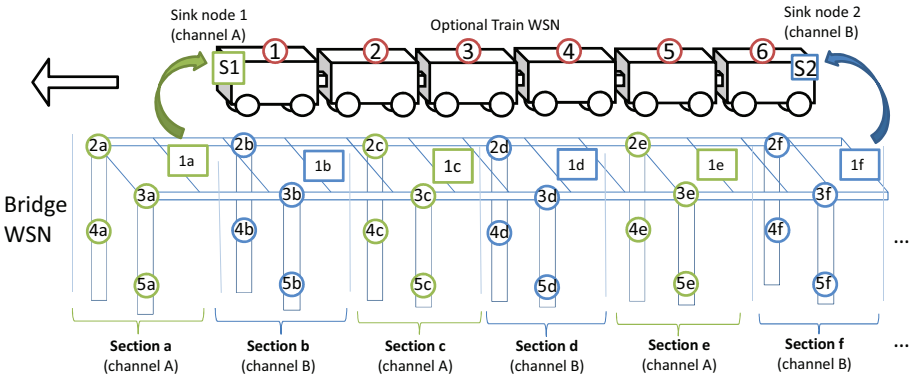


Figure 6.1: Architecture of the application prototype

of components in the system, no general testbed is mentioned in the paper. In our work we have simulated the application scenario as a whole including mobility and the mobile data transfer protocol. In addition, unlike other proposals, we make use of a middleware layer to automatically handle QoS requirements and to simplify the task of developing the application. Finally, other proposals use an application specific design whereas the use of PS-QUASAR allows us to have a more generic design. This in turn, allows us to add more nodes to the WSAN, for example nodes to cover new sections of the bridge, without having to reprogram already deployed nodes.

6.3 RAISE architecture

The general architecture of the application is depicted in Figure 6.1. The application scenario consists of a WSAN deployed on a railway bridge (referred to simply as the bridge WSAN for the rest of the chapter) and sink nodes deployed on the trains, passing through, which will collect the information sensed by the bridge WSAN. This WSAN gathers important data about the structural health of the infrastructure such as vibrations and strain. An optional WSAN could also be deployed inside the train to monitor abnormal situations as the train travels over the railway bridge or for the whole itinerary (train WSAN). This information (vibration, temperature, material deformation,...) on the carriages' health can be tracked to detect problems in the train.

In the bridge WSAN a set of different nodes are deployed along the railway infras-

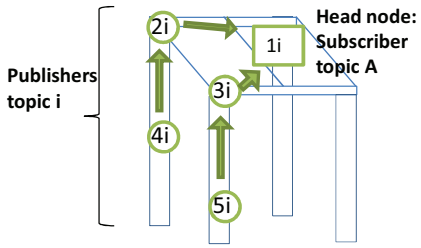


Figure 6.2: Organization of a network section: a single head node collects the information sent from other nodes

tructure. Note that nodes are not only deployed on the railway tracks but also inside the structure itself so infrastructure aging and possible incidents can be detected. The goal of the network is to self organize, to sense data whenever a train passes by and to use the next train as a data mule to upload the sensed data. The data sensed at the bridge is transferred to the train by means of sink nodes, labelled S1 and S2 in Figure 6.1.

The first application prototype was developed with a single WSAN that contained all the sensor nodes in the bridge. The tests carried out in this application prototype showed disappointing results in terms of reliability. In this first approach, a single node acts as head node of the network and collects the information that is sent by the rest of nodes. Since the reliability is significantly affected by the distance between source and destination and by neighborhood traffic, in this scenario where a single WSAN contains all nodes, collisions are frequent. As a result, reliability was shown to be around 70%-80% in our preliminary tests.

In order to increase performance, clustering techniques need to be used. Nodes along the bridge are divided into independent sections (labeled as Sections a, b, c, ... in Figure 6.1). Consecutive sections operate on different channels, namely channel A and B, so there is no interference between them. In our prototype only two channels have been used, but a greater number of channels could be used if a higher throughput is desired in the data muling process, as explained in Section 6.3.3. The use of separate sections reduces the maximum distance between nodes and the network traffic thereby improving network energy consumption and reliability.

For each Section i of the bridge WSAN, node $1i$ subscribes to information on topic S_i. The rest of the nodes in Section i publish information on topic S_i. Figure 6.2 shows

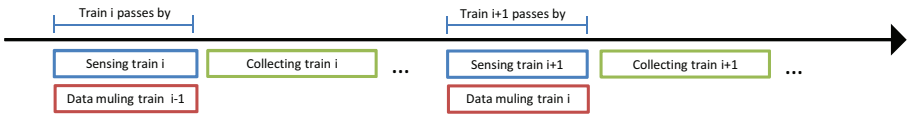


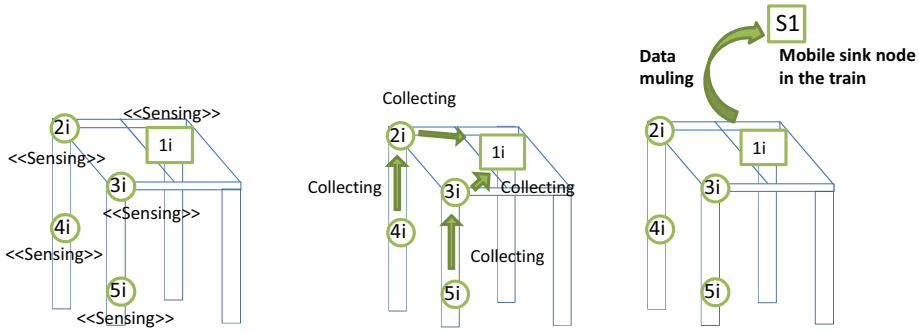
Figure 6.3: Train schedule and the execution of the different modules

the connections between the nodes in a section (determined by the node range and the location where they are deployed). In the case study prototype, each section is composed of a total of 5 nodes. All these nodes (including the head node) participate in sensing data but only the head node communicates with the train to upload the sensed data. This organization of the section (tree-based) has been chosen because it minimizes neighborhood interference and therefore improves the reliability of each section. Application developers are not directly aware of the routing protocol, nor the network organization, that is, the middleware automatically delivers the information. This allows them to add or remove nodes from each section on-the-fly, even in other topologies distinct from the tree one used in this prototype. The sensed information collected in the head nodes is stored until the next train passes by. In that moment, the data muling protocol will start uploading the information to the train.

Overall the application has three different modules: sensing module, collecting module and data muling module. Figure 6.3 shows the relationship between each of the modules and the trains's schedule. Sensing and collecting modules run on all nodes in each section whereas the data muling module is only used in head nodes. These three modules are explained in Sections 6.3.1, 6.3.2 and 6.3.3 and depicted in Figures 6.4(a), 6.4(b) and 6.4(c), respectively.

6.3.1 Sensing

The sensing module retrieves data as trains pass by. It is far more useful to gather the data when the train is passing through as this provides real information on how the infrastructure behaves when it is actually in use. This can be used to detect abnormal vibrations or material deformation which indicates that the health of the infrastructure has been compromised. The frequency rate at which to sample depends on the information to be collected. Since the information gathered by each sensor is simulated in our application prototype we have



(a) Sensing module (b) Collecting module (c) Data muling module

Figure 6.4: Different modules of the application running in the nodes

assumed a sampling rate of 2Hz and a sample size of 2 bytes. Nodes in the bridge WSAN are instructed to start sensing whenever a train passes by. To do that, the application needs to identify whenever a train is approaching the bridge in order to start collecting data. In the tests presented in Section 6.4, the command to start sensing is given by the simulation script each time a train passes by. In an actual deployment there are several alternatives that can be used to detect a nearing train. BriMon [124] for example, suggests the use of frontier nodes which are nodes placed upstream of the sensor network to detect nearing trains in time to notify the rest the network to start sensing. Another option would be to use accelerometers to detect vibrations coming from approaching trains. In the same way, when a train leaves the bridge, nodes are instructed to stop sensing.

All information sensed in each node is stored in a data cache. This data cache is accessed by the collecting module.

6.3.2 Collecting data

The application collects data by default when there are no trains on the bridge. The collecting module in each node sends the information stored in the local data cache to the head node of the section. For example, in Figure 6.2 all nodes send data packets, containing the sensor readings, to node 1 whenever there is data in the local data cache. To do this, sensor nodes call the *publish* primitive of the PS-QUASAR middleware and it automatically handles the delivery. Packet payload is filled with as much data as possible, from the local data cache in order to minimize the number of packets to be sent. In order to be sure that data is

```
Sensing module

Every SENSING_PERIOD:
    Sample s = get_sample_from_sensors();
    save_to_data_cache( s );
```

```
Collecting module – head node
At initialization
   Ps_subscribe( topic_i, handle_data_listener );

Function handle_data_listener( Address src_address, Sample data){
   save_to_data_muling_cache( data );
}
```

```
Collecting module – sensor node
Every COLLECTING_PERIOD:
   if( data_cache_not_empty ){
       Sample sample_collection[] = get_samples_from_data_cache();
       ps_publish( topic_i, sample_collection, qos_reliable );
   }
```

Figure 6.5: Pseudocode of collecting and sensing modules for Section *i* of the application scenario

delivered, communication is reliably configured. This is achievable by using an additional parameter in the *publish* primitive that accepts QoS requirements. The collection module is programmed to use retransmissions with ACKs in order to confirm that data has been delivered. When the module is enabled, data cache is periodically checked and if it contains something then a packet is sent to the head node. The data collected by each section will not be transferred until the next train arrives so packet delay is not a concern. Therefore, data in the cache does not need to be sent immediately to the head node. For each node, a collecting period of 1 second has been chosen between consecutive transmissions of data to the head node. Figure 6.5 shows the pseudocode of the sensing and collecting data modules, respectively, using the API shown in Figure 4.2. Head nodes call the *ps_subscribe* method to express their desire to receive all the information associated with a topic. Nodes in the same section use the *ps_publish* method to send the information on that same topic. The third parameter of the method establishes that the information needs to be sent reliably. The middleware automatically delivers the information to the corresponding subscribers.

6.3.3 Data muling

Once the information has been collected in the head node the next train that passes by will be used as the data mule to get the information from it. This module only runs on head nodes which are the only nodes that communicate with the train. The module is executed concurrently together with the sensing module in the head nodes. The module basically starts sending data packets from the data muling cache to the train whenever a passing train is detected.

Two issues need to be tackled in the mobile data transfer. The first one is to reliably send the information to the train and the second to send it at a speed that allows all sensed data to be uploaded to a single train. The first one is solved by using reliable transmission based on ACKs. The second depends on different parameters such as the train speed and the hardware used to transmit the data (node radio range, data rate, ...). In order to further increase the throughput of the proposed data muling protocol multiple sink nodes are used. In our prototype two sink nodes have been used in order to double the transfer rate of the protocol but a higher number of sink nodes could be used if necessary, for example if the sampling rate required is higher. The radio range of each head node is not assumed to be higher than the one for normal sensor nodes. In Figure 6.1, for example, sink node S1 collects data from head nodes 1a, 1c and 1e while sink node S2 does the same from head nodes 1b, 1d and 1f. The results presented in Section 6.4.2 show that the proposed data muling protocol is feasible.

6.4 Evaluation

The complete case study application has been implemented with the settings described in Section 6.4.1. The results in terms of reliability and quantity of data generated by the WSAN are shown in Section 6.4.2. The mobile data transfer results are discussed in Section 6.4.3. Finally the power consumption is presented in Section 6.4.4.

6.4.1 Environment set-up and scenario settings

The application scenario has been implemented in C programming language for the Tmote-sky motes running the Contiki operating system [118]. Table 6.1 shows the main features

Attribute	Value
Processor	MSP430 8MHz
Radio	CC2420 802.15.4 compliant
Battery	2 AA batteries
Power consumption	Sending: 59.1 mW Receiving: 52.2 mW CPU: 5.4 mW LPM: 0.1635 mW
Operating system	Contiki OS

Table 6.1: Tmote-sky specifications

of these motes. Power consumption in the table, and in the rest of the tests, has been calculated using the energest module [129] provided by Contiki OS. The resulting code has been simulated using the Cooja simulator [119]. The Cooja simulator emulates Tmote-sky motes at machine code instruction set level. The communication model takes into account packet loss when nodes are transmitting at the same time, namely collisions are taken into account in the simulation. The Contiki test editor plugin has been used to control the simulation and to actually simulate the movement of the train. This plugin allows users to control many different settings of the scenario, such as node position, at different instants of time. This feature has been used to actually recreate the movement of the train passing through the bridge WSAN. Nodes have been deployed as depicted in Figure 6.1, that is 30 nodes divided into 6 sections of 5 nodes each. The script simulates 20 trains passing over the bridge, one every 60 seconds. Each train moves at a speed such that the sink nodes on the train are in range of each head node for around 7 seconds. For example, this means that if both head nodes and sink nodes in the train have a radio range of 50 metres the train is is moving at a speed of 100 km/h (assuming ideal conditions).

6.4.2 Reliability and data generated by the bridge WSAN

Results obtained in the tests are summarized in Table 6.2. One section generates on average 706 bytes everytime a train crosses the bridge. The total amount of sensed data has been

Attribute	Value
Sensing rate	2Hz
Sample size	2 bytes
Mean data generated for each train in one section	706 bytes
Data collection reliablity achieved	100%
Data collection mean number of retransmissions	1.076
Data collection maximum number of retransmissions	7

Table 6.2: Data generated and reliability

received by each head node which gives a reliability of 100%. Although the maximum number of retransmissions carried out by a sensor is 7 the mean number of retransmissions is 1.076 which means that almost no retransmissions have been carried out. Also, it shows that even in networks with low traffic it is really difficult to obtain 100% reliability without using techniques such as retransmissions. This leads us to believe that simulators which do not take collisions into account do not produce realistic results.

6.4.3 Data muling

In the tests carried out each of the trains receives the readings sensed when the previous train was crossing the bridge. Each train is in range with each head node for approximately 7 seconds. The information sensed for each of the 20 trains in the test has successfully been received by the sink nodes S1 and S2. The reliability achieved is 100% because ACKs have been used to confirm the reception of data packets. To do that the runicast library provided by Contiki OS has been used. The data muling transfer rate from the bridge WSAN to nodes S1 and S2 for each train is 0.668 Kbps and 0.665 Kbps. That means that the mean data muling transfer rate for the whole system is 1.334 Kbps. Although in the application scenario all packets have been successfully transferred to the train the data muling transfer rate is really low compared to the maximum data rate of the mote (around 45 Kbps). Several factors may have influenced this data rate drop. First, the head node also carries out the sensing task at a rate of 2Hz which slows down the data muling process. Head nodes can be programmed not to carry out sensing if a higher data rate is needed in the head nodes. Also, the operating system and the retransmission mechanism introduces some

	Head node	Normal node	Sink node
Sensing and data muling	9.392	0.971	1.812
Collecting	1.950	1.832	

Table 6.3: Power consumption (mW)

latency, especially when ACKs have not been received (the radio has to wait a predefined time if no ACK has been received before sending a retransmission). Finally, the radio range of the head nodes is assumed to be relatively short (i.e. 50 metres if the train moves at a speed of 100 Km/h). By extending the radio range of head nodes the data muling transfer rate can be easily increased.

6.4.4 Power consumption

The power consumption of each kind of node has been measured and is shown in Table 6.3. This power consumption can be compared to that presented in Table 6.1 for the different modes of the mote. Power consumption during the data collection is relatively low, 1.950 mW and 1.832 for head nodes and normal nodes, respectively. Power consumption of the sink nodes is also low, although energy consumption in sink nodes is not a concern because they can be powered as they are located on the train. During the sensing and data muling processes, head nodes have the highest energy consumption since all the information gathered by the network needs to be transmitted by them. However, this only happens when trains are crossing the bridge which constitutes a really short amount of time compared to the amount of time the head nodes are collecting information.

7

Integration of SCADA and WSANs

Supervisory Control and Data Acquisition (SCADA) systems are widely used nowadays to monitor and control dispersed hardware components in industrial settings such as power plants, electrical power grids and water treatment [130]. Often, they are used in critical infrastructures where security and safety are vital factors. For this reason, they have to comply with strict regulatory standards [102]. Traditionally, SCADA systems are deployed in a monolithic way with a central datacenter and with a large amount of wiring required to connect the different hardware elements with the datacenter [131]. Due to the restricted access to these wired elements, traditional security provision focuses primarily on physical protection measures [132]. Recently, the use of WSAN technology has been acknowledged as promising for the CIP field[133]. In addition, the ability to extend the radio range through multi-hop communications can be used to extend the reach of CIP systems by monitoring more points of interest in wider deployment areas. However, in order to take advantage of WSAN, traditional SCADA acquisition processes need to change in order to accommodate this new technology. For example, secure ways of delivering infor-

mation from the sensors to the centralized SCADA must be implemented. Also, SCADAs normally integrate proprietary protocols, which make the integration with a general purpose WSAN unstraightforward, requiring the use of gateways. To tackle all these issues and to propose a generic solution to the integration problem between WSANs and SCADA was, in fact, one of the goals of the European project WSAN4CIP [64], in the context of which the solution presented in this chapter has been developed. The WSAN4CIP project globally aimed to protect Critical Infrastructures by means of a secure WSAN system architecture. In this chapter we present the solution developed for the SCADA-WSAN integration. The solution is based on the use of open protocols that work over conventional networks, such as the Internet. More specifically, the system uses IP and web services together with an open-source and web-based SCADA called Mango [134]. Two proof-of-concept demonstrators have been deployed in WSAN4CIP, one for monitoring drinking water distribution pipelines [135] and the other for monitoring an electrical power grid [136]. This chapter only focuses on the approach followed to integrate SCADA and WSAN and the way it has been applied to the latter demonstrator, henceforth designated the EDP demonstrator, which reflects the name (EDP) of the electricity distribution company.

The structure of this chapter is as follows. Section 7.1 briefly describes the electrical power grid use case where the system has been demonstrated. In Section 7.2 the SCADA-WSAN integration solution is described. For the sake of clarity, more details about the use case is given in each subsection.

7.1 Case study: An electricity distribution network

The case study used to test the integrated solution consists of a monitoring and surveillance application for an electricity distribution network located in Setúbal, Portugal. The system described has been fully implemented and used in a real demonstrator. The electrical power grid of the distribution infrastructure is schematically shown in Figure 7.1. The infrastructure principally consists of a set of substations, Medium Voltage (MV) power lines connecting substations to Medium Voltage/Low Voltage (MV/LV) power transformers residing in the secondary substations and LV power lines from the secondary substations to the customers. Some industrial customers may also get direct MV power lines. The SCADA system is, in this use case, located within one of the substation premises, in the

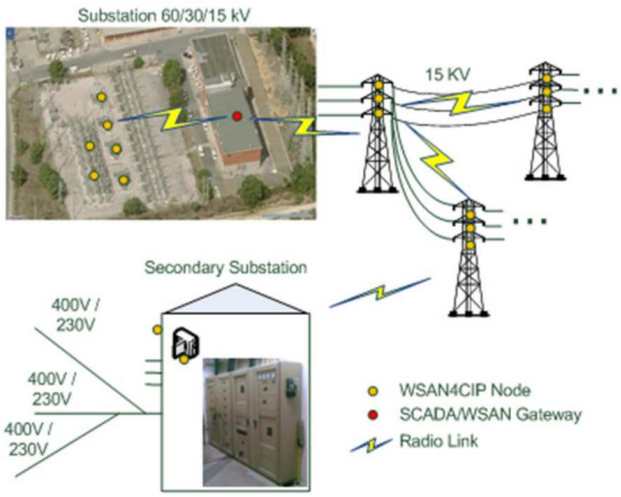


Figure 7.1: Electrical power grid distribution infrastructure

same building of the SCADA/WSAN gateway.

The WSAN based system collects information such as current and temperature values, battery level, trip coil status, and periodically sends the information to the SCADA. The SCADA shows all the collected information and allows the SCADA operator to trigger on-demand and periodical operations to query parameters in real-time. The SCADA can be accessed anywhere from a simple browser and a user account. Some of the parameters that can be controlled/monitored through the SCADA are:

1. Substation circuit breaker trip coil status;
2. Temperature of the substation power transformer oil, substation neutral reactance oil and substation neutral resistor coil box;
3. MV and LV power line current activity;
4. MV/LV power transformer hotspot detection;
5. Human activity in the secondary substation through the use of movement detectors and video cameras, in conjunction with a light actuator.

All the monitored parameters and images are visualized in the SCADA system, through a special-purpose Graphical User Interface (GUI).

7.2 SCADA-WSAN Integration

We have mentioned that SCADAs used to be really complex and expensive programs with proprietary hardware, software and communication protocols. Currently, new open alternatives that make use of the common Internet protocols (TCP/IP), have proven to be a feasible solution. They offer flexibility and can work over previously existing networks and therefore can be accessed from any place where there is an Internet connection without the need to install and configure complex programs. In general terms, the architecture proposed in this section makes use of an open-source web-based SCADA, called Mango, which proves that the use of this kind of system in the CIP context is feasible, cost-efficient, easily accessible and simple to use. Regarding the SCADA-WSAN connection, the WSAN and SCADA communication has been decoupled by means of a gateway. The gateway is in charge of gathering the information from the WSAN and retransmitting it to the SCADA. Conversely, the gateway receives requests from the SCADA and processes them, disseminating the corresponding information into the WSAN. Figure 7.2 shows the general architecture of the system. The operator accesses the web-based SCADA through a conventional browser. The SCADA receives all the information from the gateway and stores it in a database. In a similar way the gateway collects all the information from the WSAN before sending it to the SCADA. The system supports the use of multiple gateways and, therefore, can receive information from a large number of disjoint WSANs.

The SCADA is described in more detail in Section 7.2.1. How to connect the WSAN running PS-QUASAR to the SCADA by means of a gateway is explained in Section 7.2.2.

7.2.1 SCADA

A web-based open source SCADA called Mango has been used. Mango provides a browser-based SCADA that allows users to monitor and control devices over multiple protocols. Mango runs in Apache Tomcat as a server application and can be accessed from any conventional browser. In a Mango SCADA the application is divided into a set of screens each of which controls a certain aspect of the monitored system. Each of these screens is a simple web page with dynamic elements that shows the status of the monitored system and allows interaction between the SCADA and the WSAN. This allows the SCADA to be used from any conventional PC with an Internet connection. Access control is managed by

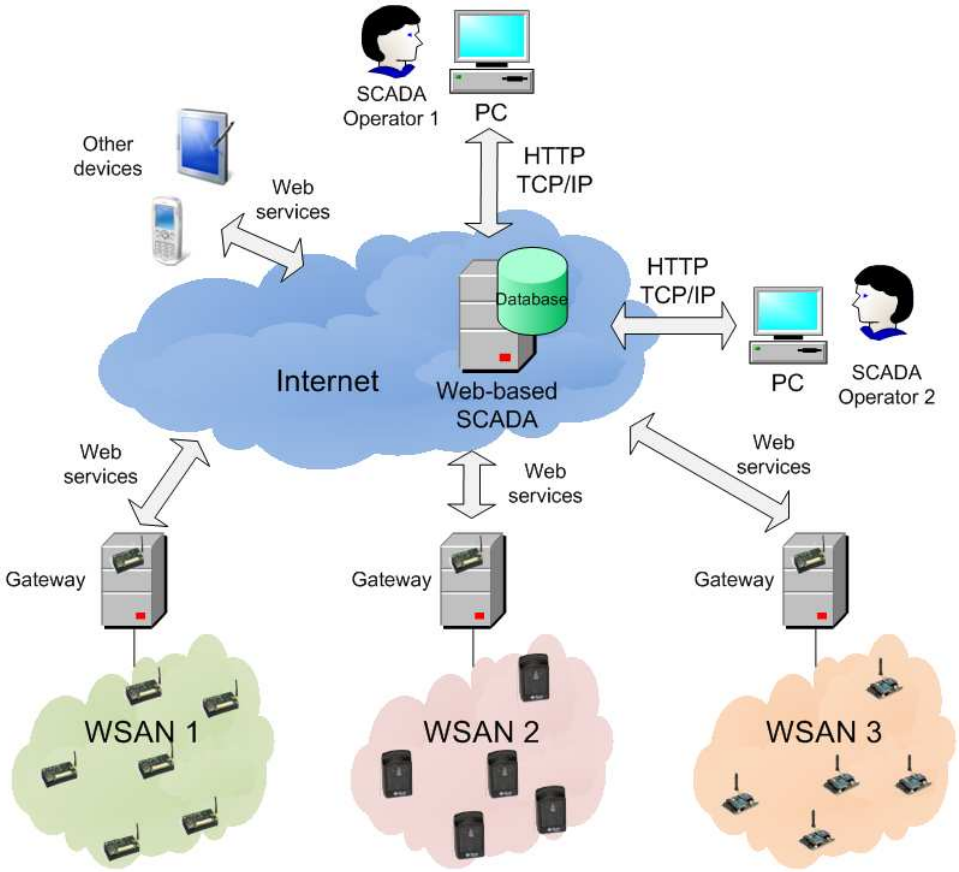


Figure 7.2: System architecture

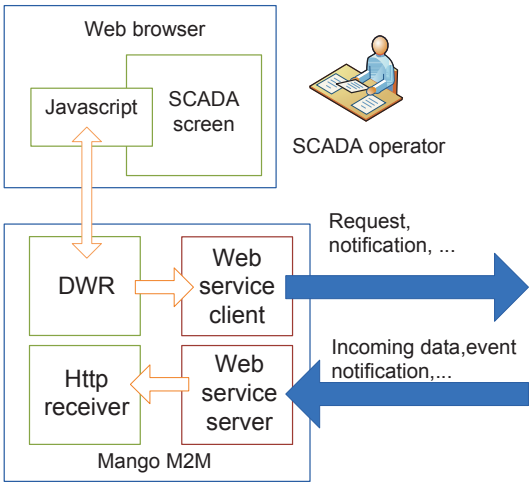


Figure 7.3: Mango-Web service communication

means of user accounts that restrict the access depending on the user role. Mango comes with predefined modules that make it possible to communicate with devices running widely used protocols such as OPC, MBus and DNP3. However, the project aimed to widen the number of devices capable of communicating with the SCADA, for example, over the Internet, and after a study of possible alternatives, web services were chosen. Web services are principally run over the well known HTTP protocol. This, together with the fact that Mango SCADA also runs over this protocol, makes it a good choice to ensure seamless interoperability with a potentially very high number of devices. In order to allow Mango to use web services a new module was developed, which is composed of a web service server to accept requests or data and a web server client to send requests and notifications. In order to secure the communication at this point the secure version of HTTP protocol (HTTPS) has been used. Also, the system uses a white list of allowed IPs in order to block access from unwanted addresses.

Figure 7.3 depicts a diagram of the Mango architecture and its relation to the web service module. The SCADA operator accesses the SCADA by means of a browser and logs on with an account that has administrator privileges. The operator can check the status of the different screens available or interact with them by means of dynamic elements implemented with AJAX. Interaction is handled in JavaScript and Direct Web Remoting (DWR). If the interaction involves a query to the WSAN, then it is carried out by the web service module that is called from DWR. In the same way, information is received through established web services residing in the web service server. Once one of these web services has been called, the information is relayed to Mango using an HTTP receiver. The HTTP receiver is a servlet that Mango provides and allows information to be noted using POST or GET methods. Once the information has been noted, Mango automatically updates the screens of all connected clients. In addition, due to the requirements of the use case, real-time streaming video was integrated inside Mango. Real-time streaming video is received from cameras placed in the secondary substation. Data is transmitted in real-time and sent directly to the Mango screen. In Mango, a java applet was integrated in the SCADA screen. Using javascript and the web service architecture mentioned before the real-time streaming can either work in on-demand or in detection mode. In on-demand mode the user must activate the control to obtain real-time images. In detection mode, the applet automatically launches whenever an alarm or abnormal situation is detected.

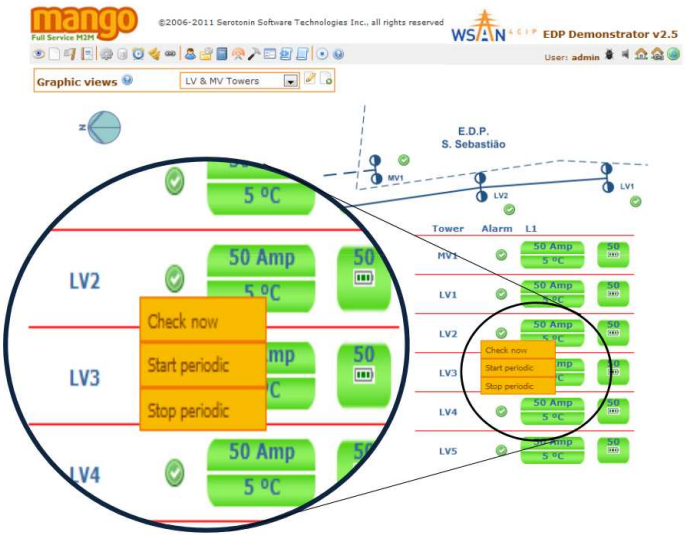


Figure 7.4: Tower monitoring SCADA screen of the EDP demonstrator

Figure 7.5: Video control and intrusion detection SCADA screen

Figure 7.4 shows one of the screens of the SCADA GUI that monitors the status of a power transmission line from the case study scenario. For each of the towers available in this section of the transmission line the following parameters are measured: current, temperature and battery level. Each of these parameters can be either provided periodically or on-demand. If any of these parameters is not within the normal operation ranges then an alarm is issued. The green color means that the respective elements respect the normal value ranges. Finally, historical data for each of these parameters can be queried. Figure 7.5 shows the video monitoring screen where real-time streaming is received from cameras deployed in the scenario. The SCADA GUI in the EDP demonstrator is composed of a total of seven screens. Security at this point is controlled in three different ways. First, only users with accounts on the system can access the SCADA. There are different types of accounts depending on the privileges needed: administrator, operator and guest. Also, the IP addresses with access to the system can be filtered to restrict access to a specific range of computers. Finally, all communications between server and users accessing the SCADA are ciphered as they run over HTTPS.

7.2.2 SCADA/WSAN Gateway

The architecture relies on the existence of a series of gateways that manage the collection of data from the WSAN and relay notification messages and events to specific sensors. The term “gateway” is usually used to refer to resource rich devices and does not have the limitation of current sensor devices. However, in this section the term is used differently. Gateway refers to a device that can handle web services both as a client and as a server. It is therefore not necessary to use energy consuming and expensive equipment as gateways. For example, in [137] the use of web services in a conventional wireless sensor is discussed. In our integration system the gateway acts as a device that allows the information to leave the WSAN to enter the internet domain and viceversa. Web services are used as the standard technologies that all gateways and SCADAs can understand and that allows seamless connectivity between all devices. Web services can be used from many different sources such as mobile phones, tablets, PCs, electrical appliances, etc, to query information about the WSANs and also to trigger operations or raise events on each WSAN.

In certain applications, the use of a single gateway is a concern as attacks at this point of the network could result in the WSAN being unable to communicate with the internet

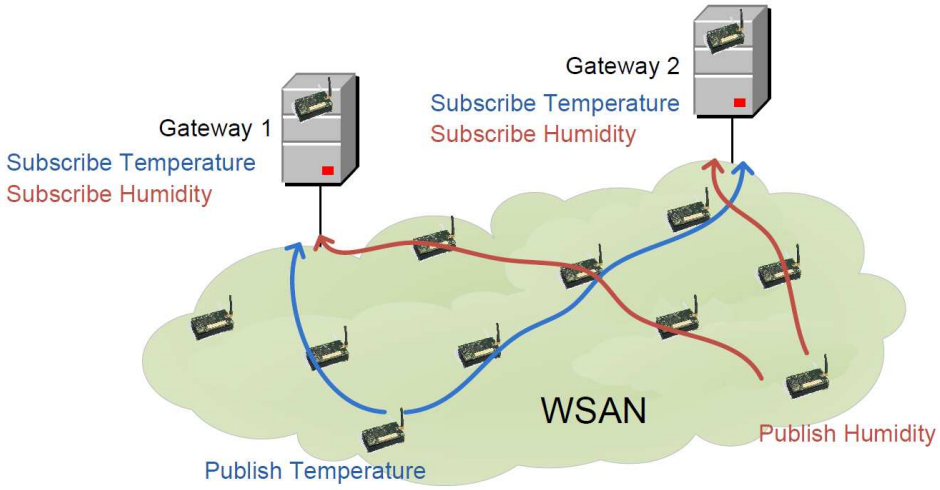


Figure 7.6: Using PS-QUASAR to report to multiple gateways

and viceversa. However, the proposed architecture together with PS-QUASAR is capable of reporting to multiple gateways. The publish/subscribe paradigm that PS-QUASAR uses together with the multicast mechanism matches the one-to-many pattern necessary to carry out this communication. Figure 7.6 shows the use of PS-QUASAR together with multiple gateways. From the point of view of the sensors there is no difference between reporting to one or more than one gateway. Once a gateway subscribes to a certain topic it will start receiving data from the publisher(s) that produces data of that same topic. This means that multiple gateways can be used to improve the system's fault tolerance. Also new gateways can be easily deployed when an existing one fails. In the same way, new nodes can join existing networks and start publishing information about existing topics.

In the case study presented in Section 7.1 the web services allow the notification of readings and alarms. Also, they handle on-demand and periodic requests to/from the gateway. Table 7.1 shows a summary of the web services used in the case study. Services offered in the SCADA are used to notify and send asynchronous readings to the SCADA such as alarms and events. On the other hand, web services offered in the gateway are used to query the WSAN status and also to carry out configuration tasks. For example the "Intrusion alarm notification" web service is used by the gateway to notify the SCADA of an intrusion that has been detected whereas "Intrusion detection activation/deactivation" is used by the SCADA to activate or deactivate the intrusion detection system.

Web Service	**Web Services offered by the SCADA System as server**	**Web Services offered by the SCADA Gateway as server**
Trip Coil test	Asynchronous test result notification	Test command Periodic test activation/deactivation
Temperature Reading	Asynchronous reading notification	Reading command Periodic reading activation/deactivation
On-demand Surveillance Video Feed	-	Surveillance video activation/deactivation
Surveillance Video Camera Control	-	Surveillance video camera control commands
Intrusion Detection	Intrusion alarm notification	Intrusion detection activation/deactivation
On-demand IR Video Feed	-	IR video camera activation/deactivation
Hotspot Detection	Hotspot alarm notification	Hotspot detection activation/deactivation
Periodic Tower Current Measurement	Asynchronous current reading	Periodic current reading activation/deactivation
Network Status Monitoring	-	Network status reading request

Table 7.1: Web services and their location

8

Conclusions and future work

In this chapter the main contributions and results of the thesis are presented. Conclusions and future work are presented in Section 8.1 and 8.2 respectively. Some general comments about the thesis and this are of research are expressed in Section 8.3. Acknowledgements are presented in Section 8.4.

8.1 Conclusions

In this thesis a set of current challenges and open issues that have prevented WSANs from becoming a mainstream technology have being identified. In particular, solutions for several issues have been tackled:

1. The low level of abstraction with which WSANs are programmed
2. The need for a routing protocol that allows nodes to exchange messages and can deal also with QoS requirements.

3. An analysis of the performance of common queue-based priority mechanisms. This study aims to identify the importance of matching network level capabilities to data link layer capabilities and comments the existence of a noticeable tradeoff between reliability and priority.

4. A way to interconnect WSANs to the internet and with different devices, to store the information the network collects and to represent it.

In order to tackle the first two challenges a middleware layer, called PS-QUASAR, has been developed. PS-QUASAR is a lightweight middleware that provides a topic-based publish/subscribe scheme for WSANs. It provides a simple publish/subscribe API that simplifies the task of developing applications for WSANs and make it less error-prone. PS-QUASAR allows multiple subscribers and publishers to coexist in the same network. It also provides a set of mechanisms that allows deadline, reliability and priority to be handled by means of a simple publish/subscribe programming model. It has been observed that the network traffic significantly degrades the node performance by causing packet collisions. A multicast mechanism has been presented that improves the energy consumption and network traffic by not duplicating packets unless the network topology requires it. The results obtained show that PS-QUASAR successfully identifies suitable paths towards each of the subscribers and can effectively handle QoS.

A railway infrastructure health monitoring application has been used to validate PS-QUASAR. It also deals with some interesting aspects such as reliability requirements and mobile sink nodes. The case study consists of a WSAN that is deployed on a railway bridge that collects information about the structural health and behavior of the infrastructure when a train travels along it and relays the readings to a set of head nodes. Head nodes then use the next train(s) as a data mule to upload the information. The WSAN makes use of PS-QUASAR to significantly simplify the task of developing the application and to allow new nodes to be added on-the-fly. Other techniques used to minimize packet loss, mainly due to collisions, are packet caching, data fusion and clustering. The evaluation carried out shows that the mobile data transfer is actually feasible and that the results obtained are satisfactory, both in terms of reliability and power consumption.

An analysis of the performance of the MAC and network layer has been carried out. In particular, the effectiveness of a priority queueing system (PQS) as a way to provide

priority has been studied. The network performance has been tested with different network topologies and sizes in order to study the effect of the sampling rate on reliability and delay. Furthermore, the impact of implementing a PQS at different communication levels (network and data link layers) to analyze the reliability and the end-to-end delay of priority packets has been studied.

The results show that a PQS should not be implemented solely at the network layer level and the importance of matching network layer and MAC layer functionality. Secondly, a PQS improves the reliability and the delay of priority packets with regard to the non-priority packets only as long as queueing exists and this only happens when there is moderate or intense traffic in the network. In addition, the network topology and its size has a direct effect on the network's performance. Thirdly, the delay of the priority packets is not always better than the delay of non-priority packets. When the queues of the data link layer only contain priority packets due to discarded packets, the average delay starts to increase as all the new priority packets that arrive at the nodes will have to wait for the queue to be processed before they can be dispatched. Finally, the use of PQS in networks with moderate traffic has an influence on reduction of delay whereas in networks with intense traffic it principally influences reliability rather than delay. In any case, in CSMA-based data link layers there is an important tradeoff between reliability and delay.

Finally, a SCADA system/web services architecture is presented to deal with the integration problem. The SCADA system uses WSANs to visualize and monitor the status of different systems. The system is entirely based on open source technologies and has been tested in the context of a European project called WSAN4CIP to monitor a power distribution infrastructure. The system integrates seamlessly with the internet and can be accessed from any conventional web browser. The system has been tested by means of a real deployment of sensors to monitor and surveil an electricity distribution network.

8.2 Future work

This section describes possible improvements and questions that are still unanswered.

- The PS-QUASAR routing protocol is based on periodic beaconing. The beaconing period is currently a predefined value that can be adjusted before deployment. However, it would be useful to be able to adapt the beaconing period to the status of the

network. In other words the beaconing period would be automatically changed by the middleware based on the existence of changes in the network topology.

- DDS is a standard specification of a publish/subscribe communication system for distributed systems used by many different devices. It would therefore be beneficial to interconnect PS-QUASAR and DDS. One possible solution is to use the SCADA infrastructure here presented to carry out the protocol translation by means of a web service.

- Collisions and medium contention are the main causes of the loss of reliability we have found in the tests presented in this thesis. Therefore, we would like to study in more detail, ways of mitigating their effects. For example, by proposing improvements over the wireless sensor MAC protocol.

- The results obtained in the evaluation of the case study (Chapter 6) with the application prototype suggest that the application scenario is actually feasible. However, there are still open questions that need to be tackled, such as which specific sensors to use in the sensor nodes and how the way in which they are deployed can affect the accuracy of the readings. There are also several issues and behaviors that have not been captured by the simulators such as the influence of the bridge's infrastructure or the speed of the train on the performance of the sensor radio that require further consideration.

8.3 General comments on this area of research

This last section concludes the thesis. In it, I wish to summarize some key points I would like to share and which I have learnt much more about the years I have dedicated to this research. These last words express my own ideas and opinions on this research topic.

The first idea deals with the use of simulators and the difficulties a developer of WSAN applications has to face. Wireless sensors are embedded devices. This together with the problems intrinsic to distributed systems makes developing WSAN applications a hard task. The existing tools and simulators are not yet fully developed commercial tools and debugging these kinds of applications is one of the most challenging tasks a developer has to

face. There are a large number of wireless sensor simulators and sometimes it is a difficult decision which developing platform to use to validate research experiments. I would like to encourage new researchers to this field to choose wireless sensor platforms which allow applications to be accurately simulated. For example, researchers are advised not to consider simulators which do not take into account collisions or that do not accurately simulate the MAC layer, as collisions and medium contention are a really important and often overlooked source of packet loss.

The second and final idea I would like to comment on concerns in the intense research that is being done on WSAN technology. Several decades has passed since the first research on this subject began and still the short/mid term prospects of this technology are unknown. Despite the abundance of protocols, platforms, programming paradigms the use of sensors is still done the old way, that is, programming specific applications to meet the requirements of specific scenarios. Furthermore, most applications that used WSANs are developed in the context of research projects or experimental set-ups. WSAN development has come a long way since research first began and is currently, in my opinion, mature enough to be used in a wide range of applications. It is actually our (the people that work in this research area) goal and mission to prove to everyone, the advantages and capabilities of this amazing technology, to show non-experts the usefulness of these applications and the possibilities of these small devices. It is also our goal to start believing in innovation rather than pure research as the logical way to help WSANs to become what was once envisioned. What I wish to say is that maybe it is time to pay more attention to the use of existing research in solving real applications rather than developing theoretical protocols that in the end are never used.

8.4 Acknowledgments

This thesis was supported by the the Spanish grant FPU-2010 (Formación de Personal Universitario) and the following projects:

- WSAN4CIP: Wireless Sensor and Actuator Networks for the Protection of Critical Infrastructures (ITC-2007-225186).
- WiCMaS: Wireless based Critical Information Management Systems (TIN2011-23795).

- Desarrollo de Software para Redes Inalámbricas de Sensores y Actores (TIC-03085).
- MDD-MERTS: Diseño y Monitorización Dirigido por Modelos de Sistemas Empotrados y Tiempo Real (TIN2008-03107).

A

Resumen

En este apéndice, se discuten las razones que han motivado el desarrollo de esta tesis a la vez que se resumen las principales aportaciones de la misma. Así mismo, se presentan las conclusiones obtenidas y los trabajos futuros.

A.1 Introducción

Las Redes Inalámbricas de Sensores (WSNs, por sus siglas en inglés) [1] proponen un sistema innovador de monitorizar e interaccionar con el entorno. Estas redes están compuestas por un conjunto de pequeños dispositivos empotrados llamados motas (Figura A.1). Las motas tienen fuente propia de energía, pueden sentir el entorno a través de una serie de sensores y pueden comunicarse entre ellas a través de radiotransmisores. Las motas se autoorganizan automáticamente para monitorizar grandes áreas de forma autónoma. La ausencia de cables y su naturaleza autoalimentada hace de esta tecnología una tecnología con posibilidades enormes. Además, el uso de un gran número de dispositivos introduce

Figure A.1: Sensores inalámbricos de diferentes compañías

redundancia en la red que puede ser aprovechada para proporcionar una alta tolerancia a fallos. Esta tecnología permite por tanto capturar la información de entornos reales y transformarla en datos digitales que pueden ser analizados y almacenados.

Las WSNs así mismo pueden hacer uso de elementos con mayores recursos llamados actores que son capaces no solo de percibir el entorno sino de actuar sobre él. Este tipo de redes que incluye actores y sensores es lo que se denomina Redes Inalámbricas de Sensores y Actores (WSANs por sus siglas en inglés) [2].

Las primeras investigaciones sobre esta tecnología empezaron alrededor de 1980 con el programa "Distributed Sensor Networks" del DARPA [3]. Los avances en diferentes tipos de tecnologías como son los procesos de miniaturización, la comunicación inalámbrica, los sensores y la tecnología de microprocesadores dieron nacimiento a las motas. A medida que han ido mejorando estas tecnologías hemos visto una mejora en el tamaño de estos dispositivos, su capacidad de cálculo, su velocidad de comunicación, sensores disponibles, etc. Esto ha llevado a la comunidad científica a proponer y soñar con nuevas aplicaciones donde este tipo de tecnología juegue un rol principal. De hecho, esta tecnología ha sido denominada como una de las más prometedoras de este siglo y que formará parte de lo que se conoce como el "Internet de las Cosas" [5]. El "Internet de las Cosas" engloba a todo tipo de dispositivos electrónicos como PCs, electrodomésticos, móviles, dispositivos empotrados, etc (Figura A.2). Las WSANs jugarían el papel de sentidos de este enorme sistema nervioso por el cual es posible que la información sea digitalizada, analizada y almacenada.

Hay un gran número de aplicaciones donde las WSANs han sido aplicadas y sus posi-

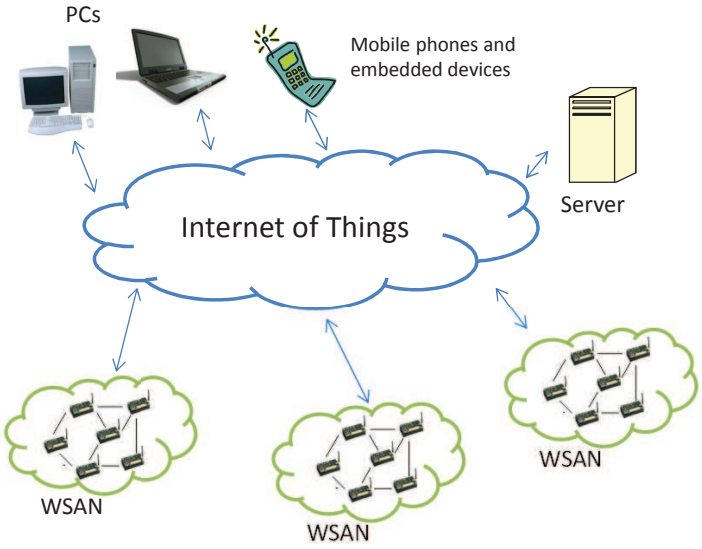


Figure A.2: Internet de las cosas (Internet of things)

bilidades han sido ya hace tiempo reconocidas en la comunidad científica. Su principal campo de aplicación es en la monitorización y control (por medio de actores) de escenarios grandes. Aplicaciones de este tipo son la detección de la polución [6][7], la monitorización de glaciares [12] y volcanes [11], la monitorización de la salud estructural de edificios e infraestructuras [15][16], la detección de intrusiones [17] o el "motion tracking" [18].

A.2 Motivación

A pesar de las altas expectativas que se han puesto en esta tecnología y de ser nombrada como una de las más importantes del siglo XXI, las WSANs todavía no se han convertido en una tecnología ubicua. Han pasado más de 30 años desde que se empezó a investigar en este campo y actualmente el número de aplicaciones reales (despliegues a largo plazo) donde esta tecnología se ha usado sigue siendo relativamente bajo. Es más, según Gartner [19], las WSANs están aun a más de 10 años de convertirse en una tecnología ampliamente establecida. ¿Qué ha sido entonces lo que ha impedido que esta transición de tecnología experimental a tecnología integrada en nuestra vida diaria se lleve a cabo? Algunos de estos factores son la falta de fiabilidad en la comunicación inalámbrica y los limitadas recursos de estos dispositivos. La WSANs constituyen un sistema distribuido y todas las dificultades

de este tipo de sistemas se dan en esta tecnología.

Además, el desarrollo de aplicaciones para WSANs es una tarea complicada y por ello no poca investigación se ha centrado en encontrar nuevos marcos de trabajo, herramientas o middlewares que proporcionen un nivel mayor de abstracción y simplifiquen la tarea de los desarrolladores [20][21]. Por otro lado, nuevas aplicaciones ideadas recientemente demandan nuevos requisitos y características que los sensores tienen que proporcionar. Por ejemplo, recientemente la posibilidad del uso de tecnología de WSANs para la protección de infraestructuras críticas ha sido calificada como prometedora [22]. En este aspecto, las WSANs tienen el potencial de formar parte integral del sistema de protección de infraestructuras críticas como pueden ser las instalaciones de generación y transmisión de energía, sistemas de telecomunicaciones, de abastecimiento de agua, etc. La naturaleza distribuida de las WSANs es particularmente apropiada frente a fallos y ataques ya que son afectados más difícilmente en su totalidad, al contrario que con los sistemas cableados.

Finalmente, las WSANs necesitan resolver el problema de la integración y representación de la información, es decir cómo hacer que la información recogida por la WSAN sea fácilmente accesible y como proporcionar a los usuarios una forma de consultar y representar esa información.

Todos estos retos pueden ser clasificados en tres categorías: abstracciones de programación de alto nivel, QoS y el problema de integración. Con respecto al primer problema, existen actualmente un conjunto de estándares para WSANs, principalmente centrados en la capa de enlace de datos como puede ser IEEE 802.15.4 o 6LoWPAN. Sin embargo, todavía no existe acuerdo acerca de que abstracción de programación usar en capas superiores a estas. Con respecto al segundo, aunque existen protocolos que permiten transmitir de forma fiable información de un nodo a otro contiguo, es necesario contar con protocolos de enrutado que se adapten a estados cambiantes en la red y posibles fallos en ella. Además, es necesario que estos protocolos proporcionen requisitos de QoS fundamentales para aplicaciones modernas como la protección de infraestructuras críticas. Por último, el problema de la integración debe idealmente ser solucionado a través de sistemas que sean independientes de la plataforma y que recojan y almacenen la información recogida por las WSANs de una forma flexible. También es necesario proporcionar formas de consultar y representar la información recogida por la WSAN en multitud de dispositivos diferentes como son los móviles, PCs, televisiones, etc.

A.3 Contribuciones de la Tesis

El objetivo de esta tesis es indagar en cada uno de los retos descritos en la sección anterior y que han impedido que las WSANs se desarrollen al ritmo esperado. En este trabajo, se analizan los problemas que tienen que ser abordados en cada uno de estos retos y se proponen soluciones a cada uno de ellos. El objetivo final es proporcionar a los desarrolladores de aplicaciones para WSANs un middleware, llamado PS-QUASAR (Publish/Subscribe QUAlity of Service Aware middlewaRe), que pueda ser usado para programar estos dispositivos de forma sencilla y menos propensa a errores. Este middleware también gestiona automáticamente requisitos de QoS especificados en la capa de aplicación. Por otro lado, los datos recogidos por la WSAN tienen que ser fácilmente consultados y almacenados.

En términos generales, esta tesis contribuye a mitigar los problemas actuales de la tecnología WSAN proponiendo para ello soluciones a diferentes niveles (capa de red, capa de aplicación y un estudio de la capa MAC). En particular estos son los puntos abordados en ella:

- Un estudio detallado de propuestas actuales de protocolos MAC, de enrutado y middlewares, haciendo especial énfasis en si son capaces de proporcionar QoS o no y su idoneidad para la protección de infraestructuras críticas. Las características deseables son identificadas así como los retos y cuestiones abiertas.

- Un protocolo de enrutado para WSANs que soporta un patrón de comunicación muchos a muchos y capaz de gestionar requisitos de QoS. Este protocolo es la base sobre la cual se sustenta una abstracción de programación basada en el paradigma publicador/suscriptor. Los requisitos de QoS gestionados por el protocolo de enrutado son fiabilidad, deadline y prioridad.

- Una abstracción de programación de alto nivel basado en el paradigma publicador/ suscriptor que simplifica el desarrollo de programas para WSANs. Los desarrolladores no tienen la necesidad de conocer la topología de la red o la forma en la que esta funciona ya que el middleware subyacente comunica automáticamente publicadores con suscriptores que han declarado su interés en la mismo tópico (del inglés “topic”).

- Un análisis del rendimiento de los sistema de prioridad basados en colas, así como un estudio de rendimiento de una WSAN bajo diferentes cargas de trabajo en términos de fiabilidad, delay y ocupación de la cola. Este trabajo hace uso del middleware PS-QUASAR y analiza diferentes topologías de red. Este estudio identifica la importancia de hacer coincidir la funcionalidad de la capa de red con la de la capa de enlace de datos para que la red se comporte de la forma esperada. También se comenta la existencia de un tradeoff importante entre fiabilidad y prioridad.

- Un caso de estudio es presentado para probar el rendimiento de PS-QUASAR. El caso de estudio consiste en una WSAN para monitorizar un puente ferroviario. La aplicación muestra maneras de hacer frente a la necesidad de recolectar grandes cantidades de datos y problemas de movilidad. En particular, los trenes que cruzan el puente son usados como "data-mule" para recoger información sobre la salud estructural de la infraestructura.

- Una arquitectura de propósito general consistente en un SCADA que conecta y recoge información de múltiples WSANs. La información es almacenada en una base de datos que puede ser consultada a través de internet usando un navegador web. Esta arquitectura es jerárquica y hace uso de software de código abierto y de tecnologías estándares como pueden ser HTTPS y servicios web.

A.4 PS-QUASAR middleware

El middleware PS-QUASAR es un middleware liviano para WSANs que proporciona una abstracción de programación para desarrolladores basado en el paradigma de programación publicador/suscriptor. Éste, hace uso un sistema basado en eventos y listeners para atender a las peticiones y tratar los datos recibidos tanto de forma local como remota. El middleware está compuesto por diferentes módulos que se muestran en la figura A.3 y que son descritos a continuación.

A.4.1 Abstracción de programación (API)

Como hemos comentado anteriormente el modelo de programación ofrecido por PS-QUASAR está basado en el paradigma publicador/suscriptor, más concretamente en el "topic-based

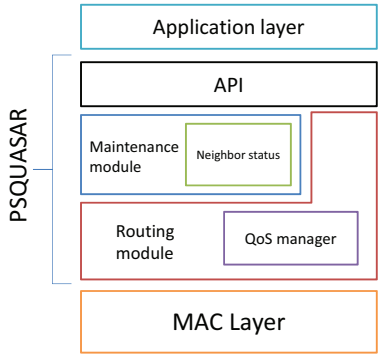


Figure A.3: Diagrama de los módulos que componen PS-QUASAR

publish/subscribe". Esto quiere decir que la comunicación entre publicadores y suscriptores solo tiene lugar si ambas partes declaran su interés en el mismo "tópico". Este tópico no es más que una cadena de caracteres elegida por el desarrollador. En el caso de PS-QUASAR y por razones de eficiencia esta cadena está codificada en una variable de 16-bits.

Creemos que este modelo es sencillo y abstrae totalmente al programador de tareas de bajo nivel como puede ser la creación de paquetes, la gestión de la radio o la de mantenimiento y búsqueda de nodos. El middleware está especialmente optimizado y diseñado para WSANs, es decir para dispositivos con baja capacidad de cálculo, con memoria limitada y comunicación inherentemente no fiable.

La Figura A.4 muestra la API proporcionada por el middleware. El modelo de programación se basa en dos mecanismos diferentes: primitivas publicador/suscriptor y listeners. Las primitivas publicador/suscriptor permiten que la información sea enviada desde publicadores a suscriptores de forma transparente. Estas dos entidades pueden estar alojadas en el mismo o en nodos diferentes. Por otro lado, los listeners son funciones que son ejecutadas cuando un mensaje o evento es recibido en un suscriptor y son usadas para iniciar la comunicación una vez que se ha producido algún evento. La Figura A.5 muestra un ejemplo del funcionamiento del sistema de comunicación basado en eventos y publicador/suscriptor. En el ejemplo, el Nodo 2 recibe una notificación de temperatura desde el Nodo 1.

A.4.2 Módulo de mantenimiento

Para que la comunicación pueda llevarse a cabo entre nodos, normalmente situados a múltiples saltos unos de otros, es necesario que los dispositivos se coordinen entre ellos de forma que

```
// SUBSCRIBER
ps_subscribe(char* topic_name, listener_function listener);
ps_unsubscribe(topic_type topic_id);
ps_unsubscribe(topic_type topic_id, listener_function listener);

// PUBLISHER
ps_publish(char* topic_name, char* data,
            unsigned short data_size, struct QoS_policy qos_policy );

Qos_policy{
    // Deadline expressed in ms
    Number deadline;
    // Indicates the number of retransmissions
    Number reliability;
    Number priority;
}

// OTHERS
ps_notify_listener(char *topic_name, char* data_to_listener, rimeaddr_t* addr);
ps_print_status();
```

Figure A.4: API de PS-QUASAR

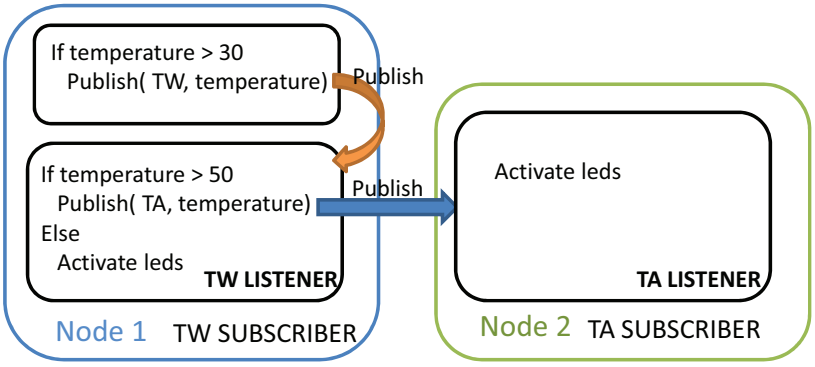


Figure A.5: Ejemplo del modelo de programación de PS-QUASAR

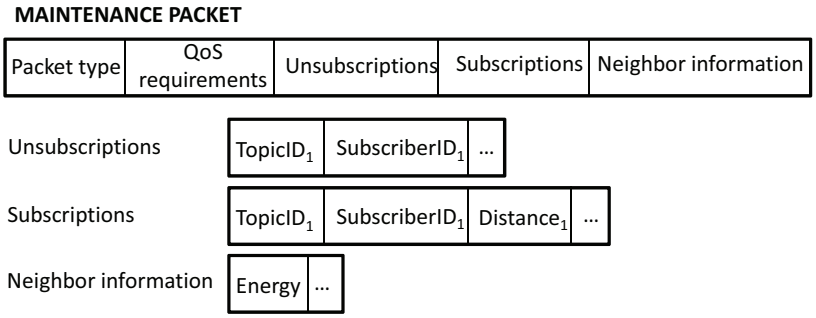


Figure A.6: Campos del paquete de mantenimiento

obtengan información de la topología de la red. Así mismo, es necesario llevar a cabo un mantenimiento de la red para detectar nuevos nodos que se añaden a ella o nodos que fallan y dejan de formar parte de la misma. Esta tarea es llevada a cabo por el módulo de mantenimiento. El protocolo de mantenimiento de PS-QUASAR está basado parcialmente en el algoritmo Bellman-Ford [109] y por tanto es susceptible de sufrir el llamado problema ´´count-to-infinity". Este identifica que bajo determinadas circunstancias donde hay nodos que fallan, el protocolo de mantenimiento puede funcionar erróneamente.

El protocolo de mantenimiento de PS-QUASAR permite conocer la existencia de cada uno de los suscriptores en la red evitando este problema y también desde cada nodo recaba cierta información acerca de los nodos vecinos que será usada por el protocolo de enrutado.

En términos simples, cada nodo informa periódicamente de su estado y pertenencia a la red mediante un paquete de mantenimiento como el mostrado en la Figura A.6. En los campos de "Unsubscriptions" cierta información es guardada para mantener la cuenta de tópicos que ya no existen o de nodos que quieren cancelar la suscripción. Esto permite resolver el problema común a los algoritmos basados en Bellman-Ford mencionada anteriormente. En los campos "Subscriptions" la información acerca de los diferentes suscriptores y tópicos existentes en la red se da a conocer a los vecinos para permitir que todos los nodos conozcan la situación de los mismos. Por último los campos "Neighbor information" contienen información útil acerca del nodo y que será usada por el módulo de enrutado para seleccionar los caminos más apropiados a cada destino.

A.4.3 Módulo de enrutado

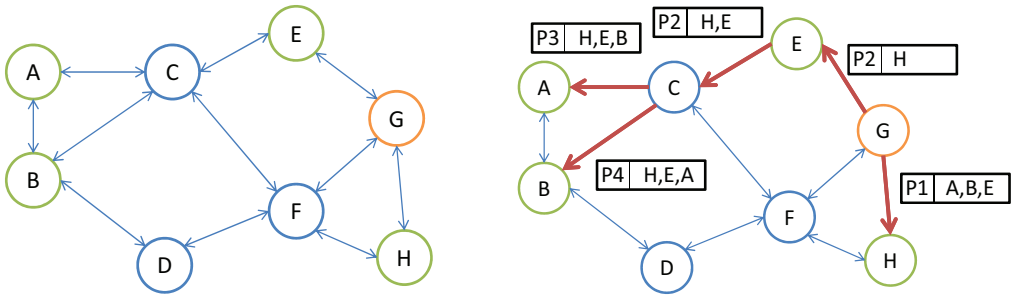
El módulo de enrutado es el encargado de transmitir los paquetes desde diferentes nodos hasta el suscriptor seleccionado. Es importante destacar que todo este proceso es transparente al usuario que solo es consciente de la comunicación a alto nivel entre publicador y suscriptor. Como hemos mencionado anteriormente el módulo de enrutado de PS-QUASAR pertenece a los protocolos de routing basados en árbol. Esto quiere decir que cada nodo en la red busca los nodos vecinos que le permitan llegar a los suscriptores existentes generando de esta forma una serie de árboles. Una característica interesante de PS-QUASAR es que hace uso de múltiples caminos hacia cada suscriptor de forma que en cada momento es seleccionado el más conveniente de acuerdo a una función peso.

$$W_i(s_j) = f_f(neighbor\_status) * f_w(neighbor\_status) \tag{A.1}$$

La función de peso, que se define en la Ecuación A.1, es el instrumento a través del cual cada nodo evalúa la idoneidad de cada vecino de ser el siguiente nodo al cual transmitir cada paquete. Es decir, esta función es usada por un nodo para evaluar todos los posibles vecinos y decidir cuál es el más idóneo (la función de peso es mayor) para transmitir un paquete. Por ejemplo, siguiendo la nomenclatura de la Fórmula A.1 la función que evalúa la idoneidad del nodo s_j de ser el siguiente en retransmitir el paquete que se encuentra en el nodo s_i está compuesta de dos funciones: f_f y f_w. f_w da un valor indicativo de la idoneidad de usar ese camino basado en la información recogida por el protocolo de mantenimiento mientras que f_f es usada para filtrar caminos que no cumplan unos requisitos mínimos. Esto puede ser usado por ejemplo para indicar al protocolo de routing que no use caminos que pasen por nodos cuya energía restante sea menor a un umbral.

Una vez evaluados todos los vecinos, el paquete es transmitido por el camino cuya función de peso es mayor. La función de peso es configurable y por tanto hay múltiples características que pueden ser usadas para decidir qué camino es el mejor para llegar al destino como por ejemplo: energía restante en el nodo, LQI, tráfico, etc.

Por último, ya que cada paquete puede ser entregado a múltiples suscriptores se ha propuesto un mecanismo de multicast que reutilizada caminos comunes hacia diferentes suscriptores en la medida de lo posible. Esto quiere decir que los paquetes enviados a múltiples suscriptores no son duplicados a menos que la topología de la red lo requiera. En



(a) Escenario de prueba. Nodo G es publicador y los nodos A, B, E and H son suscriptores

(b) La duplicación de paquetes solo se lleva a cabo cuando es requerido por la topología de la red

Figure A.7: El mecanismo de multicast de PS-QUASAR

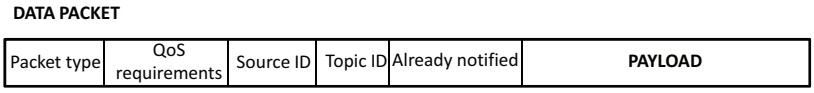


Figure A.8: Campos del paquete de datos

la Figura A.7 se muestra un ejemplo donde el mecanismo multicast se usa. Por ejemplo, el nodo H envía un paquete al nodo A y a B. El mecanismo multicast hace que el paquete inicial solo se duplique una vez que llega a C, ya que es estrictamente necesario debido a la disposición de la topología. De esta forma el número de paquetes redundantes en la red y la energía consumida general es menor. La información acerca de la distancia a la que está cada suscriptor es usada para gestionar este mecanismo de multicast.

El contenido de los paquetes de datos se muestra en la Figura A.8. La cabecera es usada para gestionar los requisitos de QoS descritos a continuación y también para gestionar el mecanismo de multicast.

El API de PS-QUASAR permite a los desarrolladores especificar una serie de requisitos de QoS que serán tenidos en cuenta por el middleware para proporcionarlos en la medida de lo posible en el envío de paquetes. Los parámetros que se pueden especificar son fiabilidad, deadline y prioridad. La fiabilidad es tratada mediante un sistema de retransmisión con confirmación nodo a nodo. El número de retransmisiones puede ser elegido por el desarrollador. La prioridad se gestiona mediante un sistema de prioridad basado en colas (que es analizado en la sección A.5). En él, los paquetes son ordenados por prioridad de forma que los paquetes más prioritarios son atendidos con más urgencia. Por último el deadline indica

Topología	Tasa de envío (ms)	Nodos fuente	Ancho de banda (kbps)
Lineal pequeña	300	2	5,76
Lineal grande	600	4	5,76
Árbol pequeño	300	3	8,64
Árbol grande	600	5	7,3
Malla pequeña	400	4	8,64
Malla grande	400	4	8,64

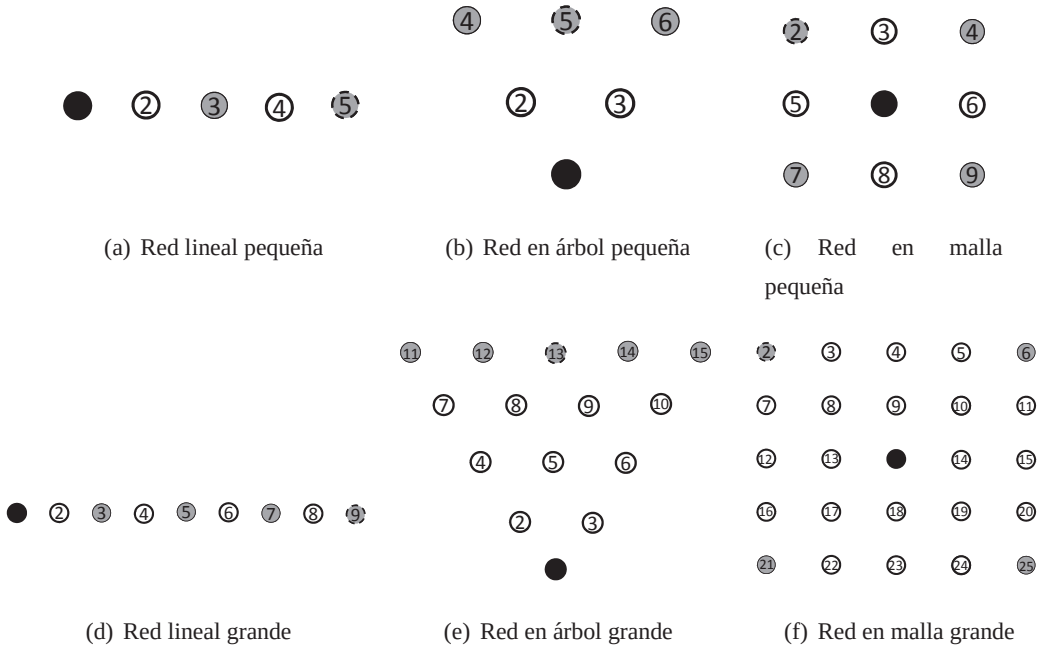
Table A.1: Ancho de banda y tasas de envío a las que la congestión empieza a ser apreciable

un tiempo máximo de envío para los paquetes tras el cual no tiene sentido que el paquete llegue a destino. El deadline es inicialmente guardado en un campo del paquete de datos y actualizado cada vez pasa por un nodo para tener en cuenta el tiempo que ha pasado desde que se mandó. Los paquetes son ordenados primero por prioridad y después por deadline en el sistema de gestión de prioridades de forma que los paquetes con un deadline menor son atendidos antes. Por último, un paquete cuyo deadline ha expirado es descartado.

A.5 Combinando la capa MAC con la capa de red

Hemos presentado un middleware que reside sobre la capa de enlace de enlace de datos y que proporciona una serie de facilidades al programador. Sin embargo, para que las características proporcionadas en la capa de red sean efectivas tiene que existir una capa MAC subyacente que sea compatible con las mismas. Para estudiar el comportamiento de la capa MAC y ver la importancia de esta combinación de capa MAC y red hemos analizado el comportamiento de la red bajo diferentes cargas de trabajo y también hemos estudiado la efectividad del sistema de prioridades basado en colas con respecto a la capa en la cual está implementada. Este estudio se ha realizado sobre diferentes tamaños y topologías de red. En este resumen, cada una de las pruebas se ilustra por medio de una o varias topologías en concreto. Las pruebas completas pueden ser encontradas en la Sección 5. Las redes usadas para las pruebas son las que aparecen en la Figura A.9. Todas las pruebas están realizadas en Contiki OS [118] y el simulador Cooja [119].

La primera serie de pruebas está destinada a descubrir cómo se comporta la red bajo

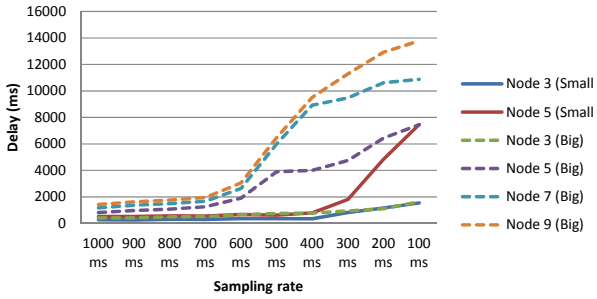


(a) Red lineal pequeña (b) Red en árbol pequeña (c) Red en malla pequeña

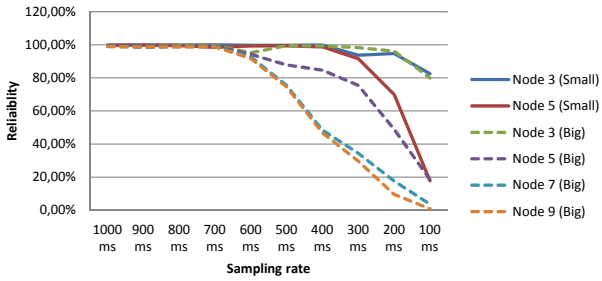
(d) Red lineal grande (e) Red en árbol grande (f) Red en malla grande

Figure A.9: Topologías de red. Negro: nodo sink. Gris: nodo fuente. Blanco: router. Borde discontinuo: nodo envía paquetes prioritarios)

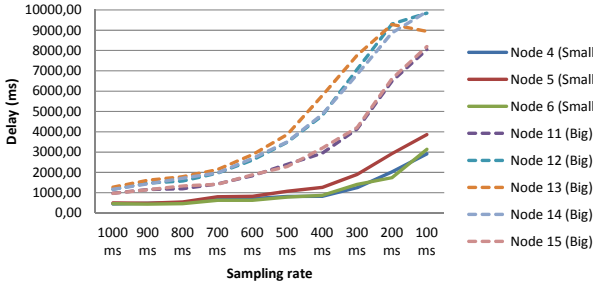
diferentes cargas de trabajo. Para ello, se han usado diferentes tasas de envío de datos en los nodos marcados como “Sources” en la Figura A.9. Cada uno de estos nodos envía un total de 1000 paquetes al nodo sink. Para cada uno de los escenarios se ha medido la fiabilidad, el delay y otras variables como el índice de ocupación de la cola MAC y el número de colisiones. En la figura A.10 podemos observar los resultados para algunas de estas topologías. Observando las figuras en general se observa que existe un valor de tasa de envío umbral para cada topología tras el cual la fiabilidad empieza a caer de forma significativa a la vez que el delay aumenta rápidamente. La Figura A.1 muestra las diferentes tasas de envío a las que esto ocurre, las cuales están muy por debajo de la capacidad de comunicación máxima de los sensores que se sitúa en torno a 40 Kbps. Se observa que la distancia entre nodos fuente y nodo sink parece influir de forma notable en el rendimiento. Así mismo, el rendimiento de los nodos que se encuentran en vecindarios más grandes (mayor número de vecinos a distancia de un salto) decae a un ritmo mayor. Estos resultados son conocidos



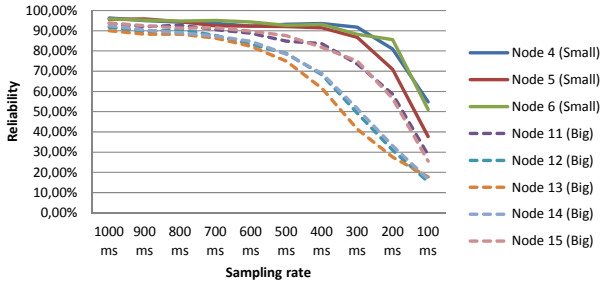
(a) Latencia de las topologías lineales



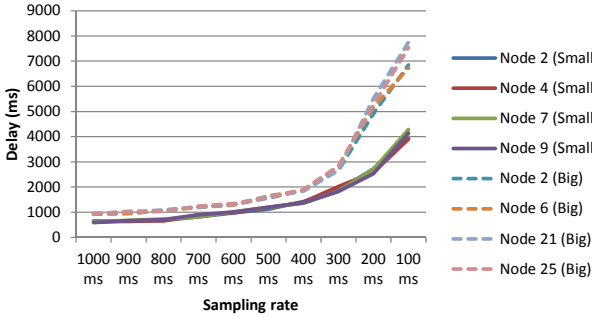
(b) Fiabilidad de las topologías lineales



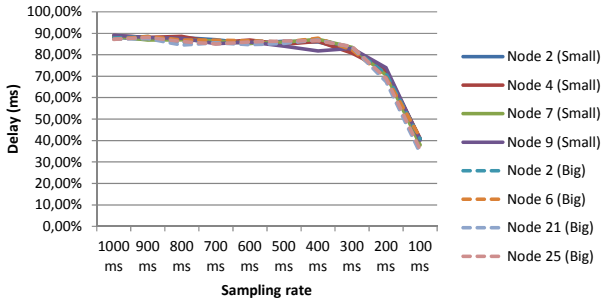
(c) Latencia de las topologías en árbol



(d) Fiabilidad de las topologías en árbol

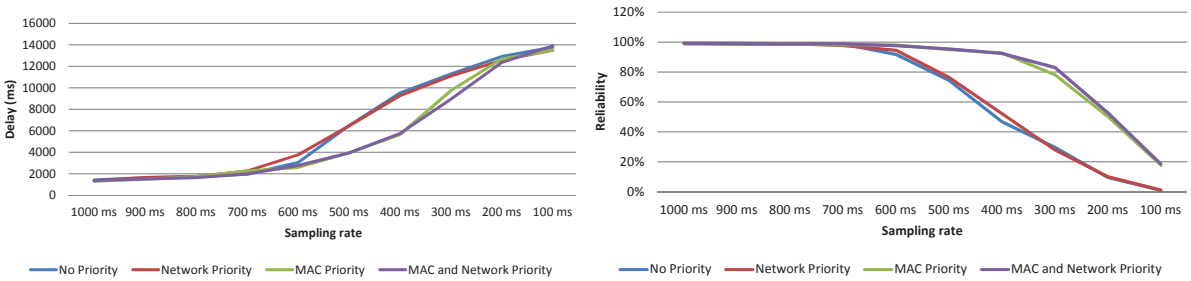


(e) Latencia de las topologías en malla



(f) Fiabilidad de las topologías en malla

Figure A.10: Resultados de rendimiento (fiabilidad y latencia)



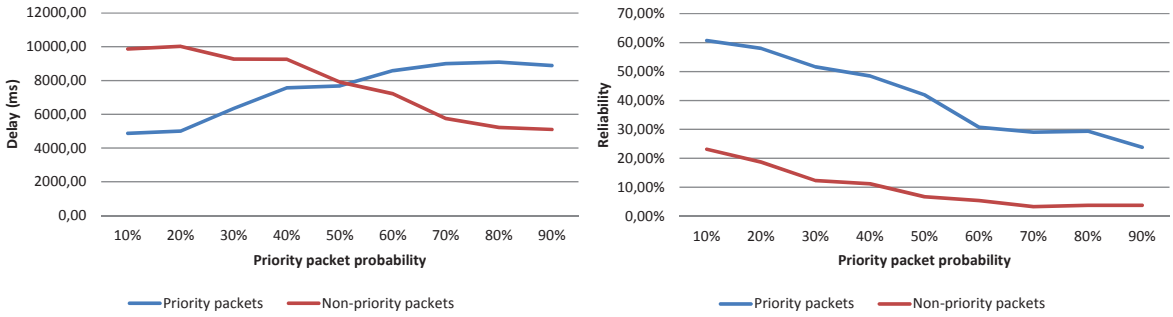
(a) Delay

(b) Fiabilidad

Figure A.11: Latencia y fiabilidad para el nodo prioritario para la red lineal grande con 4 configuraciones de prioridad diferentes

en los sistemas distribuidos y en el caso concreto de la WSANs pueden ser aplicados en la siguiente regla: mientras más pequeña y menor densidad de nodo tenga la red, mejor se comportará. Esto quiere decir que es preferible aplicar técnicas de "clustering" a usar una red única con un gran número de nodos. La pérdida de paquetes, al contrario de lo que se suele pensar, se debe a descartes de paquetes en los nodos fuente (los paquetes no llegan a ser enviados) o nodos cercanos a estos. La congestión ocurre por tanto en estos puntos y no siempre en los nodos más cercanos a los nodos sink como tradicionalmente se intuía.

La segunda prueba tiene en cuenta estos primeros resultados y consiste en el envío de 1000 paquetes por cada nodo fuente con una tasa de envío de 200 ms. Todos los paquetes enviados por el nodo marcado con borde discontinuo en cada una de las topologías de la Figura A.9 son marcados como prioritarios. Esta tasa de envío ha sido elegida en base a los resultados de la prueba anterior que indican que la red se congestiona bajo estas condiciones. Este escenario obliga al sistema de gestión de prioridades basado en colas a entrar en funcionamiento pues recordemos que solo es efectivo si hay encolamiento de paquetes en los nodos. La Figura A.11 resume los resultados de la prueba para la red lineal grande donde se han probado cuatro configuraciones distintas. En la primera configuración no existe ningún sistema de gestión de prioridades. En la segunda y tercera el sistema de gestión de prioridades está situado únicamente en la capa de red y de enlace de datos respectivamente. Por último en la cuarta el sistema de gestión de prioridades se sitúa tanto a nivel de red como a nivel de enlace de daos La principal conclusión de esta prueba es que colo-



(a) Latencia (nodo 13, tasa de envío 200ms)

(b) Fiabilidad (nodo 13, tasa de envío 200ms)

Figure A.12: Latencia y fiabilidad en base a la proporción de paquetes en la red para la red grande en árbol

car el sistema de gestión de prioridades a nivel de red sin que exista por debajo una capa MAC que pueda manejar prioridades no mejora sustancialmente el rendimiento en términos de fiabilidad y delay. Sin embargo, si este sistema está alojado en la capa MAC es posible apreciar una mejora considerable en fiabilidad y una reducción del delay medio. El rendimiento de la configuración que gestiona la prioridad a nivel de red y nivel de enlace datos simultáneamente no mejora sustancialmente con respecto al que solo gestiona la prioridad a nivel de enlace de datos. Esto se debe a que el procesamiento a nivel de red es extremadamente rápido y por tanto impide que la cola a nivel de red crezca.

Esta prueba, resalta la importancia de la elección de una capa MAC que sea capaz de gestionar la prioridad si queremos que el middleware sea efectivo en este aspecto ya que es normalmente en esta capa donde los paquetes se encolan y esperan a que el medio esté libre antes de poder ser enviados.

La última prueba llevada a cabo trata de comprobar la efectividad del sistema de gestión de prioridades (en la capa MAC) con respecto a la proporción de paquetes prioritarios en la red. Para esta prueba, cada uno de los nodos fuente envía 1000 paquetes con una tasa de envío de 200 ms. Cada nodo envía paquetes prioritarios con una determinada probabilidad. La prueba se ha repetido para diferentes valores de probabilidad y para cada topología de red. Los resultados en términos de delay y fiabilidad para la topología grande en árbol se resumen en las Figuras A.12(a) y A.12(b) respectivamente. Los resultados indican que la

tasa de fiabilidad es siempre mayor para los paquetes con prioridad y que esta tasa disminuye conforme la proporción de paquetes prioritarios en la red empieza a aumentar. Sin embargo, los resultados obtenidos de delay muestran un comportamiento en principio poco intuitivo. En líneas generales, el delay cuando la proporción de paquetes prioritarios es pequeña es menor para los paquetes prioritarios. Sin embargo, a medida que aumenta esta proporción el delay fluctúa llegando a ser mayor para paquetes prioritarios con respecto a los no prioritarios en algunos escenarios. Este comportamiento se justifica mediante dos razones principales. La primera es que un aumento de fiabilidad en paquetes prioritarios hace que haya más tráfico en la red y por tanto afecte negativamente al delay. Por otro lado, cuando el tráfico en la red es intenso y las colas de paquetes se llenan, los paquetes menos prioritarios empiezan a ser descartados en favor de los prioritarios. Esto hace que los paquetes menos prioritarios descartados no se tengan en cuenta para el delay mientras que los paquetes prioritarios sí.

La serie de pruebas realizas en este capítulo han permitido extraer una serie de conclusiones interesantes. La primera es que un sistema de gestión de prioridades solo es efectivo si se produce encolamiento lo cual ocurre cuando la congestión de la red es moderada o alta. La segunda idea que es para que cierta funcionalidad como la prioridad sea efectiva la capa MAC tiene que ser similar en funcionalidad a la capa de red. La última idea muestra que existe un tradeoff importante entre fiabilidad y delay. Un sistema de gestión de prioridades basado en colas en redes altamente congestionadas principalmente permitirá obtener una mejora en fiabilidad a costa de un aumento del delay.

A.6 Caso de estudio: Monitorización de infraestructuras ferroviarias

En esta sección el middleware PS-QUASAR es evaluado mediante un caso de estudio consistente en una aplicación de monitorización de infraestructuras. En concreto, una aplicación de monitorización de la salud estructural de un puente ferroviario.

El uso de WSANs en el contexto de esta aplicación permite instalar un sistema de monitorización permanente de costes reducidos y con la flexibilidad propia de esta tecnología.

El sistema general se muestra en la Figura A.13 y consiste en una red de sensores

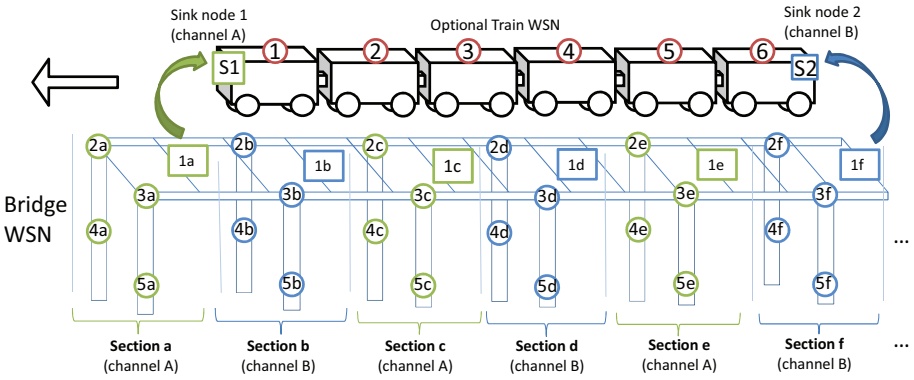


Figure A.13: Arquitectura del caso de estudio

desplegada a lo largo de un puente ferroviario. La red tiene la misión de recoger datos acerca de las vibraciones y posibles deformaciones del puente. El principal interés está en analizar el comportamiento de la estructura mientras está siendo usada, por tanto los datos se recogen principalmente mientras los trenes están cruzando el puente. Los requisitos de la aplicación es la recolección de una gran cantidad de datos de forma fiable durante periodos ocasionales (cuando el tren está usando la infraestructura) y el envío de los mismos a un centro de control remoto.

Para cumplir estos requisitos hay tres cuestiones principales que han sido abordadas:

1. Cómo organizar la red
2. Cómo recoger la información y almacenarla
3. Cómo llevar la información hasta el centro de control remoto

Una primera aproximación para abordar la primera cuestión consiste en desplegar los sensores a lo largo de la infraestructura de forma uniforme formando una gran red de sensores. Sin embargo, las primeras pruebas organizando la red de esta manera han mostrado que se obtiene una pérdida de paquetes de hasta el 30% debida principalmente a las colisiones producidas por el tráfico de la red. La causa de tal pérdida de rendimiento se debe a que las comunicaciones se llevan a cabo entre nodos distantes y que al ser una red única, el tráfico en la red puede potencialmente afectar a un número alto de nodos. Es conveniente, por tanto, segmentar la red en una serie de secciones separadas como las mostradas en la

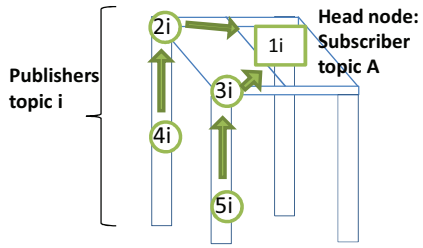


Figure A.14: Organización de una sección de la WSAN: un nodo líder recoge la información del resto de la red

Figura A.13. Cada sección trabaja de forma independiente en un canal de radio distinto. Con ello se consigue tanto disminuir la distancia entre nodos como minimizar el tráfico de cada sección.

El middleware PS-QUASAR es usado para resolver la segunda de las cuestiones. Cada una de las secciones en las cuales está subdividida la red es gestionada por PS-QUASAR de forma que todo nodo dentro ella reporta datos a un único suscriptor denominado nodo líder. La Figura A.14 muestra una de estas secciones y como está configurada con el middleware PS-QUASAR. Todos los nodos publican datos que serán recibidos por el nodo 1 que hace de nodo líder ya que es el único suscriptor en cada sección. La comunicación está configurada para hacerse de forma fiable, es decir con un número alto de retransmisiones. De esta forma se simplifica significativamente la tarea de configurar la red y se consigue que añadir nuevas secciones o nodos individuales en cualquier sección sea sencillo y fácil puesto que es gestionado automáticamente por el middleware.

La última de las cuestiones consiste en cómo hacer llegar la información recogida en los nodos al centro de control remoto. La solución propuesta consiste en usar los trenes que cruzan la infraestructura como estaciones base para recolectar toda la información recogida en cada uno de los nodos líderes de cada sección de la WSAN. Para ello existe un protocolo de “data muling” que transmite los datos de cada sección a unos nodos sumideros situados en el tren cuando éste cruza la infraestructura. Estos nodos están identificados como nodos S1 y S2 en la Figura A.13 y recogen los datos de las secciones que usan el canal de radio A y B respectivamente.

Este caso de estudio ha sido implementado en su totalidad y simulado con Cooja [119] que permite crear scripts para simular la movilidad necesaria para imitar el movimiento del

tren. Los resultados indican que el caso de estudio es en efecto viable y que el uso del middleware PS-QUASAR simplifica enormemente la gestión de la red de sensores.

A.7 Integración de WSANs, dispositivos y SCADAs en internet

En la introducción se ha comentado que el problema de la integración trata el reto de hacer que la información recogida por la red de sensores pueda ser consultada y almacenada. Para intentar abordar esta problemática se ha propuesto una arquitectura basada en tecnologías estándares y de código abierto la cual se ha probado en el contexto del proyecto de investigación europeo WSAN4CIP [64]. El caso de estudio sobre el que se ha probado la solución consiste en una red de monitorización de una estación de distribución de energía situada en Portugal.

La Figura A.15 muestra un esquema de la arquitectura de nuestra solución de integración donde coexisten múltiples WSANSs. Cada WSAN contiene un o múltiples gateways que recogen la información de ésta y hacen de pasarela entre la red e internet. Los gateways de cada red son configurados como suscriptores de los diferentes “tópicos” que existen en la red para recoger toda la información que se genera. Esta información es consultada o transferida a otros dispositivos a través de servicio webs convencionales. Para visualizar y almacenar los datos se ha usado un SCADA de código abierto llamado Mango [134]. Mango es una aplicación servidor escrita en Java que permite acceso simultáneo a múltiples usuarios a través de internet y que es usado a través de un navegador web convencional.

Creemos que las tecnologías usadas y el modelo servidor del SCADA de código abierto hace posible la interacción entre múltiples dispositivos de una forma sencilla e independiente de la plataforma y constituye una posible forma de comunicar un gran número de dispositivos heterogéneos en lo que se conoce comúnmente como “El internet de las cosas” (Internet of things).

A.8 Conclusiones

Esta tesis identifica y aborda un conjunto de retos y de cuestiones sin resolver que han contribuido a impedir el rápido desarrollo de las WSANs y el que se conviertan en una tec-

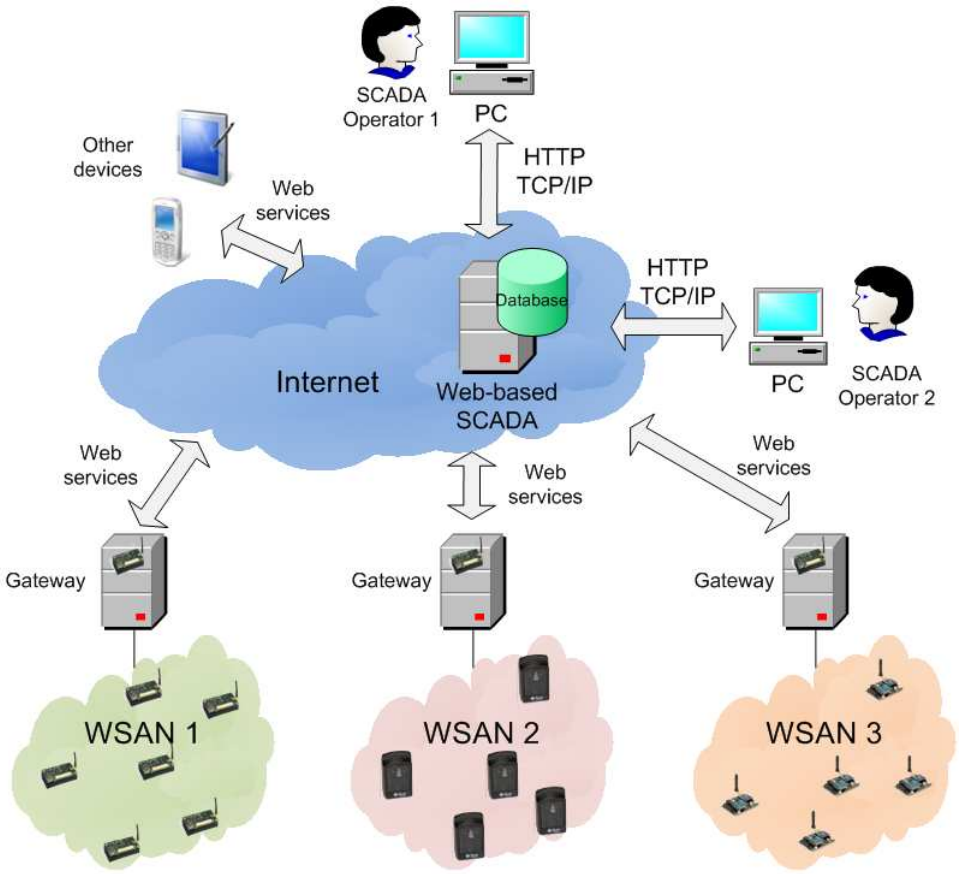


Figure A.15: Arquitectura general del sistema de integración

nología ubicua. En concreto se han propuesto soluciones a la siguiente serie de cuestiones:

1. Al bajo nivel de abstracción con la que actualmente se programa este tipo de dispositivos.

2. La necesidad de un protocolo de enrutado que permita a los diferentes nodos de una WSAN comunicarse y a su vez gestione de forma automática requisitos de calidad de servicio en las comunicaciones.

3. Una forma de interconectar diferentes WSANs, dispositivos, SCADAs en internet de forma que la información generada por todas las entidades que componen la red pueda ser compartida, almacenada y representada.

4. Un análisis del rendimiento de las WSANs que usan mecanismos de prioridad basados en colas. Este estudio identifica la importancia de la elección de una capa MAC acorde a la funcionalidad de la capa de red y destaca la existencia de un tradeoff importante entre fiabilidad y prioridad.

Las dos primeras cuestiones se abordan a través de un middleware para WSANs, llamado PS-QUASAR. PS-QUASAR es un middleware ligero que proporciona un modelo de programación basado en el paradigma de programación publicador/subscriptor. Proporciona una API de programación sencilla que simplifica la tarea de desarrollar aplicaciones para WSANs a la vez que la hace menos propensa a errores. Este middleware permite el uso de múltiples suscriptores y publicadores en una red multisalto. También proporciona un conjunto de protocolos y mecanismos para manejar y tratar requisitos de deadline, fiabilidad y prioridad de forma automáticamente y que son especificados por el desarrollador a través de la API de programación. Con objeto de aprovechar los recursos de la red y evitar en la medida de lo posible tráfico en la red se ha propuesta un mecanismo de multicast que aprovecha caminos comunes entre diferentes subscriptores y publicadores para evitar mandar paquetes duplicados mientras no sea estrictamente necesario. Los resultados obtenidos en las pruebas muestran que PS-QUASAR identifica camino apropiados para cada uno de los suscriptores y que es capaz de manejar y tratar los requisitos de QoS especificados.

Con objeto de probar el middleware, un sistema monitorización de la salud estructural de infraestructuras ferroviarias ha sido propuesto. Este caso de estudio trata aspectos interesantes como la movilidad, alta tasa de generación de datos y requisitos de fiabilidad. El

caso de estudio sobre el cual se han desarrollado las pruebas consiste en una WSAN desplegada en un puente ferroviario que es usada para recoger datos sobre la salud estructural y el comportamiento del mismo cuando el tren está haciendo uso de la misma. Los datos son almacenados temporalmente en un conjunto de nodos líderes. Los nodos líderes descargan esa información en los trenes que son usados como estaciones base móviles. La WSAN hace uso de PS-QUASAR para simplificar significativamente la tarea de desarrollar la aplicación y para permitir a nuevos nodos unirse a la red sin tener que reconfigurarla. Otras técnicas usadas para minimizar la pérdida de paquetes, principalmente debida a colisiones, es el almacenamiento temporal de paquetes, la fusión de datos y técnicas de clustering. La evaluación que se ha llevado a cabo muestra que el escenario planteado es factible, tanto en términos de fiabilidad como de consumo energético.

La tercera cuestión ha sido abordada a través de una arquitectura de integración entre WSANs, SCADAs e internet. Esta arquitectura está enteramente basada en tecnologías de código abierto y ha sido probada en el contexto de un proyecto europeo, llamado WSAN4CIP, para monitorizar un sistema de distribución de energía. Los resultados obtenidos son satisfactorios en términos de fiabilidad y versatilidad.

Por último, un análisis del rendimiento de la capa MAC y de red ha sido llevado a cabo. En particular, el estudio se centra en la efectividad de sistemas de prioridad basado en colas. Se ha analizado el rendimiento de diferentes topologías de red bajo diferentes condiciones de tráfico para estudiar el efecto de la tasa de muestreo de los sensores en la fiabilidad y el delay. También, el comportamiento de la red se ha analizado implementando el sistema de prioridad basado en colas en diferentes capas de comunicación (capa de enlace de datos y de red) para comprobar el impacto que tiene sobre la fiabilidad y el delay. Los resultados muestran que el sistema de prioridad basado en colas no debe ser implementado a nivel de red (al menos no de forma exclusiva). Si éste se sitúa en la capa MAC mejora la fiabilidad y el delay de los paquetes prioritarios con respecto a los no prioritarios siempre y cuando se produzca encolamiento de paquetes en los sensores. Además, contrario al sentido común el delay medio de los paquetes prioritarios no siempre es mejor que el de los no prioritarios. Cuando las colas de la capa de enlace solo contienen paquetes prioritarios debidos a descartes de los no prioritarios, el delay medio de estos primeros empieza a aumentar mientras que el de los segundos no. Por último podemos comentar que en redes con tráfico moderado el sistema de prioridades basado en colas influye positivamente en una

reducción del delay de los paquetes prioritarios mientras que en redes con tráfico intenso influye principalmente en un aumento de la fiabilidad y no tanto en el delay. En cualquier caso, las pruebas demuestran que existe un importante tradeoff entre fiabilidad y delay y que muchas veces no es posible mejorar ambos requisitos.

A.9 Futuros Trabajos

En esta sección se describen posibles mejoras de las diferentes propuestas presentadas en esta tesis.

- El protocolo de mantenimiento de PS-QUASAR está basado en el envío periódico de paquetes de mantenimiento. Este envío periódico actualmente es fijo y tiene un valor que puede ser ajustado antes del despliegue de la red. Sin embargo, sería interesante adaptar este periodo al estado de la red. Es decir, es el middleware el encargado de ajustar automáticamente el periodo de envío de estos paquetes ante posibles cambios en la topología de la red, por ejemplo cuando se añaden nodos o algunos nodos dejan de funcionar.

- DDS es un estándar consistente en una especificación de un sistema de comunicación basada en el paradigma de programación publicador/subscriptor. Sería interesante la idea de permitir la comunicación entre PS-QUASAR y DDS. Una posible solución es usar la infraestructura mostrada en el capítulo 7 de forma que la traducción entre ambos protocolos sea implementada en el subsistema de servicios webs.

- Las colisiones y el descarte de paquetes por contención en el medio son una de las principales causas de pérdida de fiabilidad. Sería por tanto muy interesante estudiar bajo qué condiciones ocurren e intentar proponer mejoras para mitigar los efectos de las mismas. Una forma de lograr este objetivo es proponiendo cambios o mejoras en el protocolo MAC de los sensores.

- Los resultados de la evaluación del caso de estudio presentado en el Capítulo 6 demuestran que la aplicación es factible. Sin embargo, existen todavía algunas interrogantes que necesitan ser examinadas con más detenimiento, como por ejemplo que sensores específicos necesitan ser usando en los sensores y de qué forma afecta la

forma en la que se despliegue a la precisión de sus lecturas. También existen algunos factores que no han sido tenidos en cuenta por el simulador y que requieren un estudio más detallado como por ejemplo la influencia de la velocidad del tren o la estructura ferroviaria en el rendimiento del radiotransmisor de las motas.

A.10 Comentarios generales acerca de la tesis

En esta última sección me gustaría resumir un par de ideas y de conocimientos adquiridos durante la realización de la tesis que espero que sean útiles para posteriores generaciones de investigadores. Estas ideas representan mi opinión subjetiva acerca del estado actual de la investigación en el campo de WSANs.

La primera idea tiene que ver con el uso de simuladores y las dificultades que un desarrollador de aplicaciones para WSANs tiene que afrontar. Las WSANs están compuestas de dispositivos empotrados. Esto unido a los retos y problemas intrínsecos de los sistemas distribuidos hacen que el desarrollo de aplicaciones para estos dispositivos sea una ardua tarea. Gran parte de las herramientas y los simuladores existentes todavía no son herramientas acabadas con fines comerciales y es innegable que depurar programas para WSANs es una de las tareas más complicadas para el desarrollador. Un gran número de simuladores, plataformas y sensores comerciales existen y por ello es complicado elegir o seleccionar una plataforma sobre la cual llevar a cabo las pruebas para validar trabajos de investigación. Me gustaría alentar a los investigadores a elegir una plataforma que permita simular los programas con un mínimo de exactitud. Por ejemplo, no es aconsejable considerar simuladores que no tengan en cuenta la capa de enlace de datos o las colisiones, puesto que es en estos puntos donde yacen las principales fuentes de pérdida de paquetes.

La segunda idea concierne a la intensa investigación que se está llevando actualmente en el campo de las WSANs. Ya hace un mínimo de 30 años que las primeras investigaciones de lo que hoy puede ser llamado tecnología de WSANs empezaron. Sin embargo, el futuro a medio/largo plazo de esta tecnología es aún incierto. A pesar de la gran abundancia de protocolos, plataformas, paradigmas de programación, etc, los sensores siguen usándose a la manera tradicional. Esto significa que las WSANs siguen siendo programadas a bajo nivel y cada vez para aplicaciones específicas sin hacer uso de soluciones genéricas. Además, la mayoría de aplicaciones donde se ha hecho uso de esta tecnología se

han llevado a cabo en el contexto de proyectos de investigación y no en aplicaciones reales que responden a necesidades del mercado. El desarrollo de las WSANs ha recorrido un largo camino y bajo mi opinión es una tecnología lo suficientemente madura para ser usada en multitud de aplicaciones reales de hoy en día. Es por tanto nuestra misión (de la de las personas que trabajamos en esta área de investigación o con esta tecnología) el demostrar las multitudes posibilidades de esta tecnología tan sorprendente, de enseñar a los no expertos en el área que es posible hacer uso de esta tecnología para mejorar procesos existentes y proporcionarnos nuevas facilidades.

No menos importante, es empezar a valorar un poco más la innovación y la creación de nuevas aplicaciones con tecnología existente frente a la investigación pura como forma de ayudar a las WSANs a que lleguen a ser lo que una vez se vaticinó que iban a ser. En definitiva, tal vez es hora de prestar más atención a cómo hacer uso de la investigación existente en este campo con objeto de resolver problemas reales en vez de permanecer a un nivel puramente de investigación donde gran parte de las propuestas que se hacen nunca llegan a escapar del ámbito experimental y académico.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[2] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges," *Ad Hoc Networks*, vol. 2, no. 4, pp. 351 – 367, 2004.

[3] C.-Y. Chong and S. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, pp. 1247 – 1256, aug. 2003.

[4] G. W. K. West and K. Hall., "Research and markets: Wireless sensor network," tech. rep., Technology Trends Report Q4 2010, 2010.

[5] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[6] T. L. Dinh, W. Hu, P. Sikka, P. Corke, L. Overs, and S. Brosnan, "Design and deployment of a remote robust sensor network: Experiences from an outdoor water quality monitoring network," in *Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on*, pp. 799 –806, oct. 2007.

[7] A. Khan and L. Jenkins, "Undersea wireless sensor network for ocean pollution prevention," in *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, pp. 2 –8, jan. 2008.

[8] S.-E. Yoo, J.-E. Kim, T. Kim, S. Ahn, J. Sung, and D. Kim, "A2s: Automated agriculture system based on wsn," in *Consumer Electronics, 2007. ISCE 2007. IEEE International Symposium on*, pp. 1 –5, june 2007.

[9] K. Langendoen, A. Baggio, and O. Visser, "Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture," in *Proceedings of the 20th international conference on Parallel and distributed processing*, IPDPS'06, (Washington, DC, USA), pp. 174–174, IEEE Computer Society, 2006.

[10] A. Siuli Roy and S. Bandyopadhyay, "Agro-sense: Precision agriculture using sensor-based wireless mesh networks," in *Innovations in NGN: Future Network and Services, 2008. K-INGN 2008. First ITU-T Kaleidoscope Academic Conference*, pp. 383 –388, may 2008.

[11] J. M. Lees, J. B. Johnson, M. Ruiz, L. Troncoso, and M. Welsh, "Reventador volcano 2005: Eruptive activity inferred from seismo-acoustic observation," *Journal of Volcanology and Geothermal Research*, vol. 176, no. 1, pp. 179 – 190, 2008. ¡ce:title¿Recent and active volcanism in the Ecuadorian Andes¡/ce:title¿.

[12] P. Padhy, K. Martinez, A. Riddoch, H. L. . R. Ong, and J. K. Hart, "Glacial environment monitoring using sensor networks," in *Real-World Wireless Sensor Networks*, 2005. Event Dates: June 20-21 2005.

[13] T. Gao, C. Pesto, L. Selavo, Y. Chen, J. G. Ko, J. H. Lim, A. Terzis, A. Watt, J. Jeng, B. rong Chen, K. Lorincz, and M. Welsh, "Wireless medical sensor networks in emergency response: Implementation and pilot results," in *Technologies for Homeland Security, 2008 IEEE Conference on*, pp. 187 –192, may 2008.

[14] A. Chehri and H. Mouftah, "Performance analysis of uwb body sensor networks for medical applications," in *Ad Hoc Networks* (J. Zheng, D. Simplot-Ryl, and V. Leung, eds.), vol. 49 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 471–481, Springer Berlin Heidelberg, 2010.

[15] C.-L. Fok, G.-C. Roman, and C. Lu, "Rapid development and flexible deployment of adaptive wireless sensor network applications," in *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, (Washington, DC, USA), pp. 653–662, IEEE Computer Society, June 2005.

[16] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks*, IPSN '07, (New York, NY, USA), pp. 254–263, ACM, 2007.

[17] Z. Sun, P. Wang, M. C. Vuran, M. A. Al-Rodhaan, A. M. Al-Dhelaan, and I. F. Akyildiz, "Bordersense: Border patrol through advanced wireless sensor networks," *Ad Hoc Netw.*, vol. 9, pp. 468–477, May 2011.

[18] N. Ahmed, M. Rutten, T. Bessell, S. Kanhere, N. Gordon, and S. Jha, "Detection and tracking using particle-filter-based wireless sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 9, pp. 1332 –1345, sept. 2010.

[19] M. R. Jackie Fenn, Brian Gammage, "Gartner's hype cycle special report for 2010," 2010.

[20] B. Rubio, M. Diaz, and J. M. Troya, "Programming approaches and challenges for wireless sensor networks," in *ICSNC '07: Proceedings of the Second International Conference on Systems and Networks Communications*, (Cap Esterel, French Riviera (France)), p. 36, IEEE Computer Society, Aug. 2007.

[21] L. Mottola and G. P. Picco, "Programming wireless sensor networks: Fundamental concepts and state of the art," *ACM Comput. Surv.*, vol. 43, pp. 19:1–19:51, Apr. 2011.

[22] R. Roman, C. Alcaraz, and J. Lopez, "The role of wireless sensor networks in the area of critical information infrastructure protection," *Information Security Technical Report*, vol. 12, no. 1, pp. 24 – 31, 2007.

[23] F. Xia, "Qos challenges and opportunities in wireless sensor/actuator networks," *SENSORS*, vol. 8, p. 1099, 2008.

[24] *Requirements of Quality of Service in Wireless Sensor Network*, 2006.

[25] A. Arora, R. Ramnath, E. Ertin, P. Sinha, S. Bapat, V. Naik, V. Kulathumani, H. Zhang, H. Cao, M. Sridharan, S. Kumar, N. Seddon, C. Anderson, T. Herman, N. Trivedi, C. Zhang, R. Shah, S. Kulkarni, M. Aramugam, and L. Wang, "Exscal: Elements of an extreme scale wireless sensor network," pp. 102–108, 2005.

[26] uC/OS II, "`http://micrium.com/page/products/rtos/os-ii`."

[27] ZigBee, "ZigBee Alliance, http://www.zigbee.org,"

[28] 6LoWPAN, "IETF, http://www.ietf.org/dyn/wg/charter/6lowpan-charter.html,"

[29] WirelessHART, "HART Communication Foundation , http://www.hartcomm.org,"

[30] C. H. Liu, A. Gkelias, Y. Hou, and K. K. Leung, "Cross-layer design for qos in wireless mesh networks," *Wirel. Pers. Commun.*, vol. 51, no. 3, pp. 593–613, 2009.

[31] A. Bakshi and V. K. Prasanna, "The abstract task graph: a methodology for architecture-independent programming of networked sensor systems," in *In Workshop on End-to-end Sense-and-respond Systems (EESR*, 2005.

[32] J. Paek, B. Greenstein, O. Gnawali, K.-Y. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, and E. Kohler, "The tenet architecture for tiered sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 4, 2010.

[33] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho, and M. A. Perillo, "Middleware to support sensor network applications," *IEEE Network*, vol. 18, no. 1, pp. 6–14, 2004.

[34] J. Herbert, G. Ling, and K. Fei, "Mobile agent architecture integration for a wireless sensor medical application," 2006.

[35] P. Levis and D. Culler, "Mate: A tiny virtual machine for sensor networks," 2002.

[36] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "Tinyos: An operating system for sensor networks," in *in Ambient Intelligence*, Springer Verlag, 2004.

[37] R. Müller, G. Alonso, and D. Kossmann, "A virtual machine for sensor networks," *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 3, pp. 145–158, 2007.

[38] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 122–173, 2005.

[39] C. Srisathapornphat, C. Jaikaeo, and C. chung Shen, "Sensor information networking architecture and applications," *IEEE Personal Communications*, vol. 8, pp. 52–59, 2001.

[40] R. Gummadi, O. Gnawali, and R. Govindan, "Macro-programming wireless sensor networks using kairos," 2005.

[41] R. Newton, G. Morrisett, and M. Welsh, "The regiment macroprogramming system," in *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, (New York, NY, USA), pp. 489–498, ACM, Apr. 2007.

[42] D. Gelernter, "Generative communication in linda," *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 1, pp. 80–112, 1985.

[43] C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. L. Murphy, and G. P. Picco, " TinyLime: Bridging Mobile and Sensor Networks through Middleware," in *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005)*, (Kauai Island (Hawaii, USA)), pp. 61–72, IEEE Computer Society Press, March 2005.

[44] P. Costa, L. Mottola, A. L. Murphy, and G. P. Picco, "Teenylime: transiently shared tuple space middleware for wireless sensor networks," in *MidSens '06: Proceedings of the international workshop on Middleware for sensor networks*, (New York, NY, USA), pp. 43–48, ACM, 2006.

[45] M. Kushwaha, I. Amundson, X. Koutsoukos, E. Neema, and J. Sztipanovits, "Oasis: A programming framework for service-oriented sensor networks," in *In Proc. 2nd Int. Conf. on Communication Systems Software and Middleware*, (Bangalore, India), pp. 941–947, Jan 2007.

[46] A. Rezgui and M. Eltoweissy, "Service-oriented sensor-actuator networks: Promises, challenges, and the road ahead," *Computer Communications*, vol. 30, pp. 2627–2648, 2007.

[47] E. Cañete, J. Chen, L. Llopis, and B. Rubio, "A service-oriented middleware for wireless sensor and actor networks," in *Sixth International Conference on Information Technology: New Generations (ITNG 2009)*, (Las vegas, NV, USA), pp. 575–580, IEEE Computer Society Press, CPS Conference Publishing Services, Apr. 2009.

[48] J. H. Schönherr, H. Parzyjegla, and G. Mühl, "Clustered publish/subscribe in wireless actuator and sensor networks," in *Proceedings of the 6th international workshop on Middleware for pervasive and ad-hoc computing*, MPAC '08, (New York, NY, USA), pp. 60–65, ACM, 2008.

[49] G. Russello, L. Mostarda, and N. Dulay, "A policy-based publish/subscribe middleware for sense-and-react applications," *Journal of Systems and Software*, vol. 84, no. 4, pp. 638 – 654, 2011. The Ninth International Conference on Quality Software.

[50] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, pp. 114–131, June 2003.

[51] D. A. Tran and L. H. Truong, "Enabling publish/subscribe services in sensor networks," in *Advances in Next Generation Services and Service Architectures* (R. Publishers, ed.), 2011.

[52] M. Albano and S. Chessa, "Publish/subscribe in wireless sensor networks based on data centric storage," in *Proceedings of the 1st International Workshop on Context-Aware Middleware and Services: affiliated with the 4th International Conference on Communication System Software and Middleware (COMSWARE 2009)*, CAMS '09, (New York, NY, USA), pp. 37–42, ACM, 2009.

[53] E. Souto, G. Guimarães, G. Vasconcelos, M. Vieira, N. Rosa, C. Ferraz, and J. Kelner, "Mires: a publish/subscribe middleware for sensor networks," *Personal Ubiquitous Comput.*, vol. 10, pp. 37–44, Dec. 2005.

[54] H. Cam, O. K. Sahingoz, and A. C. Sonmez, "Wireless sensor networks based on publish/subscribe messaging paradigms," in *Proceedings of the 6th international conference on Advances in grid and pervasive computing*, GPC'11, (Berlin, Heidelberg), pp. 233–242, Springer-Verlag, 2011.

[55] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo, "Fault-tolerant wireless sensor network routing protocols for the supervision of context-aware physical environments," *Journal of Parallel and Distributed Computing*, vol. 66, no. 4, pp. 586 – 599, 2006.

[56] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "Mqtt-s — a publish/subscribe protocol for wireless sensor networks," *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops COMSWARE 08*, no. January, pp. 791–798, 2008.

[57] S. Taherian and J. Bacon, "A Publish/Subscribe Protocol for Resource-Awareness in Wireless Sensor Networks," in *Proceedings of the International Workshop on Localized Algorithms and Protocols for Wireless Sensor Networks (LOCALGOS 2007)* (J. Aspnes, C. Scheideler, A. Arora, and S. Madden, eds.), vol. 4549 of *Lecture Notes in Computer Science (LNCS)*, (Santa Fe, NM, USA), pp. 27–38, IEEE Computer Society, Springer, June 2007.

[58] K. Shi and Z. Deng, "Tinymq : A content-based publish / subscribe middleware for wireless sensor networks," *Science And Technology*, no. c, pp. 12–17, 2011.

[59] M. Sharifi, M. Taleghan, and A. Taherkordi, "A publish-subscribe middleware for real-time wireless sensor networks," in *Computational Science – ICCS 2006* (V. Alexandrov, G. van Albada, P. Sloot, and J. Dongarra, eds.), vol. 3991 of *Lecture Notes in Computer Science*, pp. 981–984, Springer Berlin / Heidelberg, 2006.

[60] DDS, "Data Distribution System http://portals.omg.org/dds/."

[61] P. Boonma and J. Suzuki, "Tinydds: An interoperable and configurable publish/subscribe middleware for wireless sensor networks," in *Principles and Applications of Distributed Event-Based Systems*, pp. 206–231, IGI Global, 2010.

[62] 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), Las Vegas, Nevada, USA*, June 2004.

[63] U. S. Government, "Critical foundations, protecting america's infrastructures," tech. rep., President's Commision on Critical Infrastructure Protection, Washington, October 1997.

[64] "WSAN4CIP. Wireless Sensor and Actuator Networks for the Protection of Critical Infrastructures. EU FP7." `http://www.wsan4cip.eu`.

[65] CI2RCO, "Europen ciip newsletter," *CI2RCO (Critical Information Infrastructure Research Co-ordination)*, vol. 3, Febraury 2007.

[66] D. Bopping, "Ciip in australia," in *First CI2RCO critical information infrastructure protection conference*, (Rome, Italy), March 2006.

[67] *Wireless Sensor Networks: QoS Provisioning at MAC and Physical Layers*, 2009.

[68] I. Demirkol, C. Ersoy, and F. Alagoz, "Mac protocols for wireless sensor networks: a survey," *Communications Magazine, IEEE*, vol. 44, no. 4, pp. 115–121, 2006.

[69] C. Saul, A. Mitschele-Thiel, A. Diab, and M. Abd rabou Kalil, "A survey of mac protocols in wireless sensor networks," in *Proceedings of IWK, Internationales Wissenschaftliches Kolloquium*, 2007.

[70] *A Survey: MAC Protocols For Applications Of Wireless Sensor Networks*, 2006.

[71] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 3, pp. 493–506, 2004.

[72] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 171–180, ACM, 2003.

[73] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 95–107, ACM, 2004.

[74] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks," *Parallel and Distributed Processing Symposium, International*, vol. 13, p. 224a, 2004.

[75] S. C. Ergen and P. Varaiya, "Pedamacs: Power efficient and delay aware medium access protocol for sensor networks," *IEEE Transactions on Mobile Computing*, vol. 5, pp. 920–930, 2006.

[76] A. Warrier, J. Min, and I. Rhee, "Z-mac: a hybrid mac for wireless sensor networks," pp. 90–101, ACM Press, 2005.

[77] J. Zheng and M. J. Lee, "A comprehensive performance study of ieee 802.15.4," *Sensor Network Operations*, p. 14, 2003.

[78] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "Performance evaluation of the ieee 802.15.4 mac for low-rate low-power wireless networks," in *In Workshop on Energy-Efficient Wireless Communications and Networks (EWCN '04), held in conjunction with the IEEE International Performance Computing and Communications Conference (IPCCC*, pp. 701–706, IEEE, 2004.

[79] SunSPOT, "http://www.sunspotworld.com."

[80] N. Saxena, A. Roy, and J. Shin, "Dynamic duty cycle and adaptive contention window based qos-mac protocol for wireless multimedia sensor networks," *Computer Networks*, vol. 52, no. 13, pp. 2532 – 2542, 2008. (1) Research and Trials for Reliable VoIP Applications; (2) Wireless Multimedia Sensor Networks.

[81] H. Kim and S.-G. Min, "Priority-based qos mac protocol for wireless sensor networks," in *IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, (Washington, DC, USA), pp. 1–8, IEEE Computer Society, 2009.

[82] M. Ali, T. Suleman, and Z. Uzmi, "MMAC: a mobility-adaptive, collision-free MAC protocol for wireless sensor networks," in *24th IEEE International Performance, Computing, and Communications Conference, 2005. IPCCC 2005*, pp. 401–407, 2005.

[83] T. Watteyne, I. Augé-Blum, and S. Ubéda, "Dual-mode real-time mac protocol for wireless sensor networks: a validation/simulation approach," in *InterSense '06: Proceedings of the first international conference on Integrated internet ad hoc and sensor networks*, (New York, NY, USA), p. 2, ACM, 2006.

[84] Y. Li, C. S. Chen, Y.-Q. Song, and Z. Wang, "Real-time QoS support in wireless sensor networks: a survey," in *7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems - FeT'2007*, (Toulouse France), 2007.

[85] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, pp. 325–349, 2005.

[86] N. V. Subramanian, "Energy aware routing and routing protocol for wireless network," tech. rep., Department of Computer Science, University of North Carolina, USA, 12 2004.

[87] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *Wireless Communications, IEEE*, vol. 11, pp. 6–28, December 2004.

[88] I. F. Khan and M. Y. Javed, "A survey on routing protocols and challenge of holes in wireless sensor networks," in *ICACTE '08: Proceedings of the 2008 International Conference on Advanced Computer Theory and Engineering*, (Washington, DC, USA), pp. 161–165, IEEE Computer Society, 2008.

[89] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications*, vol. 7, pp. 16–27, 2000.

[90] T. He, J. A. Stankovic, C. Lu, and T. Abdelzaher, "Speed: A stateless protocol for real-time communication in sensor networks," 2003.

[91] E. Felemban, S. Member, C. gun Lee, and E. Ekici, "Mmspeed: Multipath multi-speed protocol for qos guarantee of reliability and timeliness in wireless sensor networks," *IEEE Trans. on Mobile Computing*, vol. 5, pp. 738–754, 2006.

[92] B. Deb, S. Bhatnagar, and B. Nath, "Reinform: reliable information forwarding using multiple paths in sensor networks," in *Local Computer Networks, 2003. LCN '03. Proceedings. 28th Annual IEEE International Conference on*, (Bonn/Königswinter, Germany), pp. 406–415, Oct. 2003.

[93] K. Akkaya and M. Younis, "An energy-aware qos routing protocol for wireless sensor networks," *Distributed Computing Systems Workshops, International Conference on*, vol. 0, p. 710, 2003.

[94] M. Younis, M. Youssef, and K. Arisha, "Energy-aware routing in cluster-based sensor networks," 2002.

[95] M. Perillo and W. Heinzelman, "Dapr: A protocol for wireless sensor networks utilizing an application-based routing cost," in *In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC*, pp. 1528–1533, IEEE, 2004.

[96] H. Chao and C. Chang, "A fault tolerant routing protocol in wireless sensor networks," *Int. J. Sen. Netw.*, vol. 3, no. 1, pp. 66–73, 2007.

[97] I. Jawhar, N. Mohamed, M. Mohamed, and J. Aziz, "A routing protocol and addressing scheme for oil, gas, and water pipeline monitoring using wireless sensor networks," in *Wireless and Optical Communications Networks, 2008. WOCN '08. 5th IFIP International Conference on*, pp. 1 –5, May 2008.

[98] M. Strasser, A. Meier, K. Langendoen, and P. Blum, "Dwarf: delay-aware robust forwarding for energy-constrained wireless sensor networks," in *Proceedings of the 3rd IEEE international conference on Distributed computing in sensor systems*, DCOSS'07, (Berlin, Heidelberg), pp. 64–81, Springer-Verlag, 2007.

[99] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He, "Rap: a real-time communication architecture for large-scale wireless sensor networks," in *Real-Time and Embedded Technology and Applications Symposium, 2002. Proceedings. Eighth IEEE*, pp. 55–66, 2002.

[100] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*, (Washington, DC, USA), p. 8020, IEEE Computer Society, 2000.

[101] I. I. S. Systems, "A strategic approach to protecting scada and process control systems," tech. rep., 2007.

[102] J. F. Keith Stouffer and K. Scarfone, "Guide to industrial control systems (ics) security special publication 800-82 (final public draft)," Gaithersburg, MD: National Institute of Standards and Technology, 2008.

[103] K. Fenrich, "Securing your control system," in *Paper presented at the 50th Annual ISA. POWID Symposium/17th ISA POWID/EPRI Controls and Instrumentation Conference*, June 2007.

[104] T. hoon Kiml, "Integration of wireless scada through the internet," *International Journal of Computers and Communications*, vol. 4, no. 4, pp. 75–82, 2010.

[105] C. Amarawardhana, K. Dayananada, H. Porawagama, and C. Gamage, "Case study of wsn as a replacement for scada," in *Industrial and Information Systems (ICIIS), 2009 International Conference on*, pp. 49 –54, dec. 2009.

[106] X. Bai, X. Meng, Z. Du, M. Gong, and Z. Hu, "Design of wireless sensor network in scada system for wind power plant," in *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, pp. 3023 –3027, sept. 2008.

[107] S. Veleva and D. Davcev, "Implementation perspectives of wireless sensor and actuator networks in smart grid," in *Advanced Materials Research*, vol. 433 - 440, pp. 6737 – 6741, january 2012.

[108] C. Alcaraz, J. Lopez, J. Zhou, and R. Roman, "Secure scada framework for the protection of energy control systems," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 12, pp. 1431–1442, 2011.

[109] A. Tanenbaum, *Computer Networks*. Prentice Hall Professional Technical Reference, 5th ed., 2010.

[110] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, (Tampa, Florida, USA), Nov. 2004.

[111] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, (Tampa, Florida, USA), Nov. 2006.

[112] J. Chen, M. Díaz, B. Rubio, and J. M. Troya", "Raise: Railway infrastructure health monitoring using wireless sensor networks," in *Proceedings of 4th International Conference on Sensor Systems and Software*, (Lucca, Italy), June 2013.

[113] A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol," Tech. Rep. T2011:13, Swedish Institute of Computer Science, Dec. 2011.

[114] J. Lloret, M. Garcia, D. Bri, and S. Sendra, "A wireless sensor network deployment for rural and forest fire detection and verification," *Sensors*, vol. 9, no. 11, pp. 8722–8747, 2009.

[115] K. Lin, "Research on adaptive target tracking in vehicle sensor networks," *Journal of Network and Computer Applications*, vol. 36, no. 5, pp. 1316 – 1323, 2013.

[116] A. Rosi, M. Berti, N. Bicocchi, G. Castelli, A. Corsini, M. Mamei, and F. Zambonelli, "Landslide monitoring with sensor networks: experiences and lessons learnt from a real-world deployment," *Int. J. Sen. Netw.*, vol. 10, pp. 111–122, Aug. 2011.

[117] M. Bottero, B. D. Chiara, and F. Deflorio, "Wireless sensor networks for traffic monitoring in a logistic centre," *Transportation Research Part C: Emerging Technologies*, vol. 26, no. 0, pp. 99 – 124, 2013.

[118] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, (Tampa, Florida, USA), Nov. 2004.

[119] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pp. 641 –648, nov. 2006.

[120] Spanish Official Bulletin of the State (BOE), "Instrucción sobre las inspecciones técnicas en los puentes de ferrocarril (instructions for technical inspections of railway bridges) (itpf-05). fom/1951/2005." `http://www.boe.es/boe/dias/2005/06/24/pdfs/A22192-22199.pdf`, 2005.

[121] M. J. Whelan, M. Fuchs, M. V. Gangone, and K. D. Janoyan, "Development of a wireless bridge monitoring system for condition assessment using hybrid techniques," in *Proceedings of SPIE, the International Society for Optical Engineering*, pp. 28–32, 2007.

[122] R. Bischoff, J. Meyer, O. Enochsson, G. Feltrin, and L. Elfgren, "Event-based strain monitoring on a railway bridge with a wireless sensor network," in *4th International Conference on Structural Health Monitoring of Intelligent Infrastructure (SHMII-4)*, July 2009.

[123] E. Aboelela, W. Edberg, C. Papakonstantinou, and V. Vokkarane, "Wireless sensor network based model for secure railway operations," *Performance, Computing, and Communications Conference, 2002. 21st IEEE International*, vol. 0, p. 83, 2006.

[124] K. Chebrolu, B. Raman, N. Mishra, P. K. Valiveti, and R. Kumar, "BriMon: A Sensor Network System for Railway Bridge Monitoring," in *The 6th Annual International Conference on Mobile Systems, Applications and Services (MobiSys)*, June 2008.

[125] R.-G. Lee, K.-C. Chen, C.-C. Lai, S.-S. Chiang, H.-S. Liu, and M.-S. Wei, "A backup routing with wireless sensor network for bridge monitoring system," *Measurement*, vol. 40, no. 1, pp. 55 – 63, 2007.

[126] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks*, IPSN '07, (New York, NY, USA), pp. 254–263, ACM, 2007.

[127] J. P. Lynch, Y. Wang, K. J. Loh, J.-H. Yi, and C.-B. Yun, "Performance monitoring of the geumdang bridge using a dense network of high-resolution wireless sensors," *Smart Materials and Structures*, vol. 15, no. 6, p. 1561, 2006.

[128] S. Kundu, S. Roy, and A. Pal, "A power-aware wireless sensor network based bridge monitoring system," in *Networks, 2008. ICON 2008. 16th IEEE International Conference on*, pp. 1 –7, dec. 2008.

[129] 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, (New York, NY, USA), pp. 28–32, ACM, 2007.

[130] Internet Security Systems, "A strategic approach to protecting SCADA and process control systems," tech. rep., 2007.

[131] B. Galloway and G. Hancke, "Introduction to industrial control networks," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 2, pp. 860–880, 2013.

[132] K. Fenrich, "Securing your control system," in *Proceedings of 50th Annual ISA. POWID Symposium/17th ISA POWID/EPRI Controls & Instrumentation Conference*, June 2007.

[133] A. Sleman and R. Moeller, "Integration of wireless sensor network services into other home and industrial networks; using device profile for web services (dpws)," in *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pp. 1–5, 2008.

[134] Serotonin Software, "Mango, Open Source M2M." `http://mango.serotoninsoftware.com`.

[135] S. Peter and G. Weber, "Securing your control system," in *Eurescom mess@ge, issue 1/2011*, 2011.

[136] A. Grilo, A. Casaca, P. R. Pereira, L. Buttyán, J. Gonçalves, and C. Fortunato, "A wireless sensor and actuator network for improving the electrical power grid dependability," in *NGI'12*, pp. 71–78, 2012.

[137] N. Y. Othman, R. Glitho, and F. Khendek, "The design and implementation of a web service framework for individual nodes in sinkless wireless sensor networks," in *Computers and Communications, 2007. ISCC 2007. 12th IEEE Symposium on*, pp. 941–947, 2007.