Università degli Studi di Ferrara

DOTTORATO DI RICERCA IN
SCIENZE DELL'INGEGNERIA

CICLO XXIX

COORDINATORE Prof. Stefano Trillo

A Functional Safety Approach to Wireless Sensor Networks for Heterogeneous Applications

Settore Scientifico Disciplinare ING-INF/05

Dottorando
Dott. Luca Dariz

_____
(firma)

Tutore
Prof. Massimiliano Ruggeri

_____
(firma)

Anni 2014/2016

Sommario

L'ampio uso di sensori in applicazioni industriali rappresenta un'opportunità di incremento di precisione, comodità per l'operatore e sicurezza. la possibilità di raccogliere dati per l'uso immediato o un'analisi a posteriori, apre nuove opportunità di business che non erano possibili fino a qualche anno fa. Per questo motivo, è sempre più frequente trovare applicazioni industriali che includano sensori di vario tipo; l'uso di una rete di sensori wireless (WSN) è in questo caso una soluzione attraente che permette di ridurre di molto i costi. In quelle applicazioni che possono essere considerate critiche per la sicurezza (safety-critical) l'uso di una WSN è stato finora scoraggiato dal fatto che il canale di comunicazione wireless non è considerato sicuro. Nonostante ciò, recenti avanzamenti tecnologici e concettuali, come ad esempio il *black channel principle*, e l'evoluzione degli standard di sicurezza funzionale, suggeriscono che questa situazione possa cambiare in futuro; perché ciò accada, è necessaria un'analisi approfondita delle implicazioni del richiedere un determinato livello di sicurezza sullo sviluppo del sistema. Con riferimento ai principali standard di sicurezza funzionale come IEC 61508, IEC 62061, ISO 13849 e ISO 25119, è possibile definire una co-progettazione hardware e software per rendere i componenti di una rete di sensori wireless adatti ad essere usati in applicazioni safety-critical.

Il lavoro descritto in questa tesi copre le tematiche tecniche riguardanti la progettazione e lo sviluppo di un nodo WSN sicuro, dagli aspetti hardware a quelli software fino ai protocolli di comunicazione. Rispetto ai protocolli real-time esistenti per WSN, questa tesi propone una versione migliorata che permette di ridurre la latenza di comunicazione nel caso peggiore, permettendo un loop di controllo più veloce oppure l'utilizzo di un numero maggiore di nodi wireless all'interno della stessa rete. Successivamente sono analizzati i problemi legati allo sviluppo del software, mostrando l'inadeguatezza dei sistemi operativi usati comunemente in ambito WSN, proponendo invece l'uso di sistemi operativi hard real-time come punto di partenza. Infine, un ulteriore contributo è dato da un'architettura hardware ottenuta applicando gli standard di sicurezza funzionale, principalmente ISO 61508 e ISO 25119. Diverse varianti sono quindi possibili, a seconda delle caratteristiche dei componenti che si intende utilizzare.

Uno dei risultati principali di questa tesi è un'implementazione prototipale, progettata presso CNR-IMAMOTER, che è in corso di realizzazione, e sarà usata in futuro per sviluppare un dispositivo proof-of-concept, usabile per realizzare un banco di prova per applicazioni WSN sicure.

Abstract

The extensive use of sensors represents an opportunity for industrial applications to become more precise, more comfortable for the operator and more safe. The possibility to collect data, for immediate use or offline analysis, is opening new business opportunities that were not possible until just some years ago. For this reason, it is becoming always more recurrent to have industrial applications with sensors involved; here, the use of a Wireless Sensor Network (WSN) is an attractive solution that would greatly reduce costs. In those applications that can be considered safety-relevant the use of a WSN is currently discouraged or prohibited because the wireless communication channel is not considered safe. However, recent technology and conceptual advances such as the black channel principle, and the recent evolution of functional safety standards, suggest that this can change in the future; in order to make this possible, a thorough analysis on the implication of requiring a certain safety level on the development of a system is required. Referring to the most used safety standards, the IEC 61508, IEC 62061, ISO 13849 and the ISO 25119 standards, it is possible to define an hardware and software co-design to make wireless sensor modules suited to be used in safety-critical applications.

The work described in this thesis cover the technical aspects of the design and development of a safe WSN node, from hardware aspects to software and protocol issues. Compared to existing real-time protocols for WSN, this thesis propose an enhancement which decreases the worst-case communication latency, allowing for faster control loops or a greater number of sensors to be employed. Software issues are then analysed, with reference to the requirements of functional safety standards, showing the inadequacy of common WSN operating systems and proposing the use of hard-real time operating systems as a starting code base. Finally, the hardware architectures proposed in this thesis are derived applying international functional safety standards, mainly ISO 61508 and ISO 25119. Different architectural variants are then possible, depending on the characteristics of the chosen components.

One of the main results of this thesis is a prototype implementation that has been designed and is being realised at CNR-IMAMOTER, and will be used to develop a proof-of-concept device which can then be used to realise a test-bed.

Contents

Chapter 1

Introduction

Industrial technologies are being revolutionised thanks to the introduction of connectivity under various forms. For example, in the automotive field, wireless remote car locking has been one of the first innovation using the idea of wireless connectivity. The availability of low-cost connectivity modules, the continuous increase of throughput on short-range (Wi-Fi) and long-range (3G-4G) radio technologies and more recently the explosion of personal hand-held devices with the computing power of a small PC, made a whole range of new applications possible. Common examples in automotive are infotainment systems on high-end machines, where one or more monitors are capable of video and audio streaming, and an in-car wireless network is available. Furthermore, remote connectivity is a reality both to provide in-vehicle connectivity to Internet and to emergency services. More recently an Internet-connected aftermarket device is used by insurance companies to monitor the habits of customers. Similar applications are found in the off-road and heavy-duty vehicles, in particular in the agricultural domain, where the typical applications are Farm Management Information Systems (FMIS). They are very powerful tools to monitor and assist the operations on the various regions of the cultivation, so the farmer can have a greater and more comprehensive control over it and increase the overall productivity. In particular, when used with the standard ISOBUS communication protocol stack (or proprietary elements offering similar functionality), they make it possible a whole range of operations in the field of precision farming, like ad-hoc treatment in different field regions.

A growing field is also that of wireless sensor networks (WSN). While sensor networks are traditionally wired, for example by means of serial links like RS485, the possibility of using a wireless link instead represents a paradigm shift. Applications that were previously seen as extremely expensive due to wiring and communication costs, for example extensive field monitoring or monitoring of urban areas, become much more cost-effective when low-power and low-rate wireless protocols were deployed, one for all the IEEE 802.15.4 standard and

the various application protocols using it, like Zigbee. The applications are so numerous that a great number of different protocol stack exist nowadays, mostly proprietary, at the point that we could list tenths of different technology and still some would be missing. However the technologies explicitly created with the intention of being a standard, like IEEE 802.15.4, are still the most common and easy to find in terms of available integrated circuits like transceivers and software protocol stacks, from different vendors. This make them the ideal technology to use as a research starting point. In fact, it is usually possible to modify almost the entire protocol stack, with the exception of the physical layer, for which at most there are a few fixed options available on a same chip. This is not true for example for IEEE 802.11, where the PHY and part of the MAC layer are tightly coupled and often implemented in the same chip. However the application fields of wireless sensor networks have been so far in the consumer market, where an hardware of software failure is generally not a critical situation. In spite of this, many WSN systems are designed to run for months or even years without human intervention like rebooting a node. The purpose of this thesis is to investigate the use of WSN in industrial fields, where requirements like safety and more generally dependability can be present. Safety-related systems usually have some real-time requirements, either because it is required to perform its normal function (e.g. tracking a movement) or because it must be diagnosable, to be able to prevent erroneous and potentially dangerous conditions when the normal communication or operation timings are not respected, usually by means of a timeout. This interest is also visible in efforts like real-time aware extensions to WSN technologies, like IEEE 802.15.4e. Software and hardware architecture also play an important role in the reliability of a WSN node, at the point that specific design and implementation techniques are required by international functional safety standards.

1.1 Organization of this thesis

In chapter 2 the main functional safety standards and architectures are analysed; this will serve as a basis for many discussions in the following chapters. This chapter is placed first because often it will be necessary to refer to it in the subsequent chapters, when referring to safer hardware architectures, protocols and software.

Chapter 3 is an overview of current vehicular network technology, with reference mainly to the heavy-duty application field, presenting an analysis of security aspects and a vision for the future evolution of vehicular network infrastructure and technology. This chapter is an example of application field where a safe WSN can be used, and it is important to identify the specific challenges that are being addressed nowadays. In particular, the evolution of vehicular networks is here proposed borrowing some concepts that have established themselves in the IT

industry.

Chapter 4 presents some current WSN technology, preparing for the discussion in chapters 5 and 6, where the main contributions of this thesis are presented. This overview is by no means exhaustive, since many domain-specific WSN technology exist, and is focused on technical aspects. The characteristics of each selected technology are briefly listed, then an example evaluation is showed, for three different sample applications.

In chapter 5 both the communication protocols at application level and the real-time aspects are analysed, proposing some enhancement to existing standards like IEEE 802.15.4. These parts of the WSN stack are usually implemented in software, so they are often fully customizable. An early implementation testing the proposed changes was realised in the context of the FACTOTHUMS project, using COTS hardware modules, and this allowed to identify at an early stage early some of the shortcomings of present architectures.

Chapter 6 presents the hardware architecture of a safe WSN node and safety considerations about the software to be used. Here the experience from the work done in chapter 5 has been fundamental for the design and implementation of a prototype WSN node, allowing to combine the work done on WSN protocols with the concepts of safe hardware and software.

Finally chapter 7 concludes the thesis, summarising the main contributions.

Chapter 2

Safe Architectures

In this chapter an overview of the relevant safety standards is given, highlighting the design principles. A specific section is dedicated to hardware, software and communication aspects, which will be applied in the following chapters.

2.1 Safety Standards and Principles

2.1.1 Functional Safety

Functional safety addresses the Electrical, Electronic, Programmable Electronic (E/E/PE) parts of a safety-relevant system. It is defined in ISO 61508-4 [1] as

> *part of the overall safety relating to the EUC and the EUC control system that depends on the correct functioning of the E/E/PE safety-related systems and other risk reduction measures*

provided that safety is defined as[1]

> *freedom from unacceptable risk of physical injury or damage*

The key concepts of functional safety are strictly related to the dependability of a system. According to [2] and [3], the objectives of functional safety are:

1. the quantification of every potential risks in terms of severity, probability, controllability for every system functionality;
2. the identification of safety mitigation measures which can reduce those risks to tolerable levels, reducing its negative impact in case of failure;
3. the validation of the functional safety measures implemented by the system.

[1] http://www.iec.ch/functionalsafety/explained/

When the context of the system changes, the safety properties also change, including those attributes, checks and mechanisms designed to mitigate the risks associated with the system. Therefore, a safe system should be designed to detect hazardous conditions and to switch the machine in a safe state, e.g. a de-energised state, in order to prevent further risks or damages.

2.1.2 ISO 61508

The ISO 61508 [1] is an A-type standard and is the root of every standard concerning the functional safety of E/E/PE systems. Being an A-type standard, it covers only basic concepts, establishing common design principles and a common terminology. It is a very generic and application independent standard; the principles and definitions it introduces are then recalled and specialised by application-specific standards, for example ISO 25119 for agricultural and forestry machines, ISO 15998 for heavy-duty machines and ISO 26262 for automotive. The standard does not only cover the requirements specific specific to E/E/PE system development, but it also encompass the management of design process, the operation and the system maintenance throughout its whole life-cycle, from concept to decommissioning. The essence is that all activities relating to functional safety are managed in a planned and methodical way, with each phase having well-defined inputs and outputs. This enables a process of verification where a check is made at the conclusion of each phase, to confirm that the required outputs have in fact been produced as planned. The ability to check (or validate) that specifications have been properly implemented throughout the safety life-cycle is one of the foundations of functional safety. The premise is that such a structured approach will minimise the number of systematic faults which are built-in to safety-related systems. The standard is composed by 7 parts:

1. General system safety requirements, documentation and safety assessment;
2. Specific requirements for E/E/PE safety-related systems, system design requirements;
3. Additional requirements for E/E/PE safety-related systems, software requirements;
4. Definitions and abbreviations;
5. Guidelines and examples for determining safety integrity levels;
6. Guidelines on the application of parts 2 and 3, calculations, modelling and analysis;
7. Techniques and measures to be used to control and avoid faults

ISO 61508 defines a Safety Integrity Level (SIL) as a performance index for safety functions. Four levels are covered, from SIL1 (less demanding) to SIL4 (more demanding), and for each one the standard details the requirements necessary to achieve it. These requirements are

more rigorous at higher levels of safety integrity in order to reduce the residual probability of dangerous failure. An E/E/PE safety-related system will usually implement more than one safety function; if the safety integrity requirements for these safety functions differ, the requirements applicable to the highest relevant SIL shall apply to the entire E/E/PE safety-related system, unless a sufficient independence between the implementation of safety functions can be proved. Finally, if a single E/E/PE system is capable of providing all the required safety functions, and the required safety integrity is less than that specified for SIL1, then ISO 61508 does not apply.

2.1.3 ISO 25119

The ISO 25119 [4] is the C-type standard for agricultural and forestry machinery. Being a C-type standard, it only applies to specific machine categories, and it either refers to general standards like ISO 61508 or define different requirements that prevail over A-type standards. The standard is composed by four parts and it defines the safety requirements for the safety-related parts of a control system. It defines a complete safety life-cycle, covering the concept phase, the system design and implementation and the operations after the start of production. In particular it requires the adoption of a V-model for the overall system design, as in Figure 2.1. The V cycle is composed of different sub-phases like definition of specifications at system/module level, implementation and testing at system/module level.
The ISO 25119 uses its own definition for measuring the safety integrity of a safety function, the Agricultural Performance Level (AgPL), ranging from AgPL A (less demanding) to AgPL E (most demanding). Each AgPL requires the combination of different requirements in terms of mean time to failure of the hardware components (MTTF), the diagnostic coverage of the different blocks (DC), common cause failure (CCF) and software required level (SRL). In particular ISO 25119 has a thorough description of the requirements for each SRL for each stage of the V-cycle.More details are given on section 2.3.

2.1.4 ISO 15998

The ISO 15998 [5] is the C-type norm specific for earth-moving machinery and it embraces the electronic control units of the machine control systems (MCS) performing safety functions. The standard gives a guidance for risk assessment and for the identification of the required SIL according to ISO 61508. It also defines a set of parameters that has to be considered in the functional safety analysis:

- Climatic conditions (i.e. temperature, humidity)
- Mechanical condition (i.e. vibration, shock)

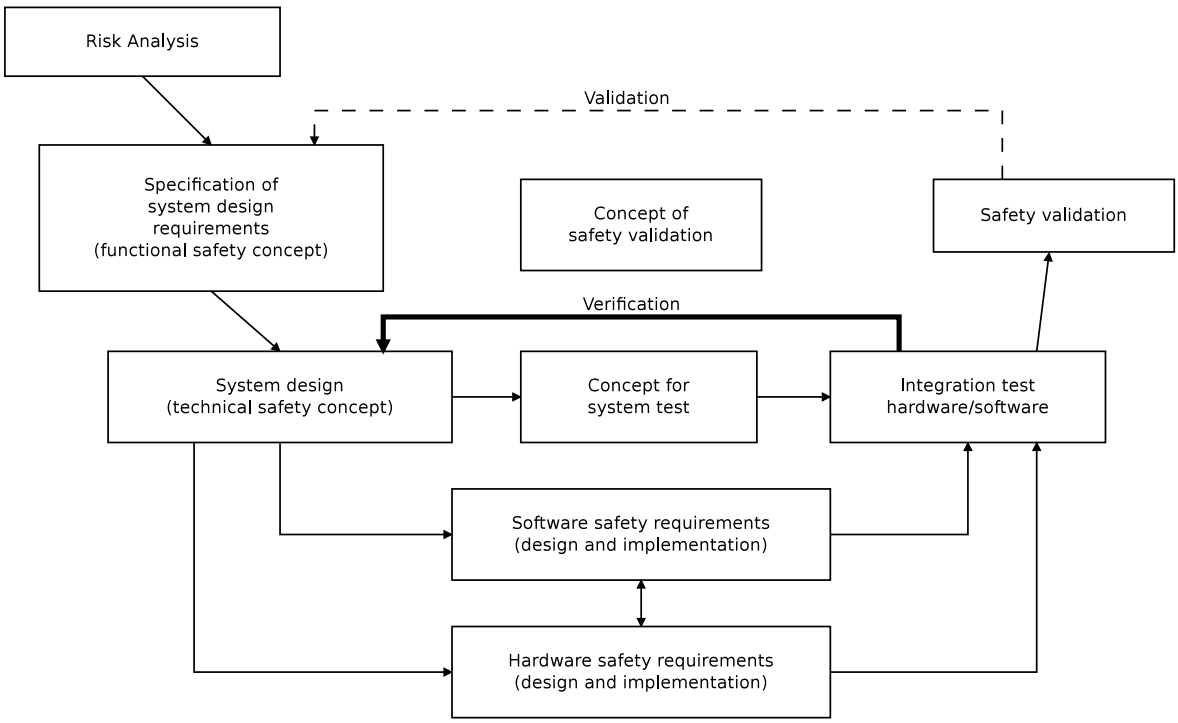


Figure 2.1: V-model for the overall system design, redrawn from ISO 25119.

- Corrosion conditions (i.e. salt spray, gas pollution)
- Electromagnetic compatibility
- Power source voltage fluctuation.

ISO 15998 also gives some examples for the hazards analysis of some safety functions such as steering, braking, propelling and operating. For MCS with SIL ≥ 1, the manufacturer must document system fault detection and tolerance mechanisms. A safe-state must be achieved in case a malfunction or failure is detected on a safety-critical MCS, providing reduced system performance or an alternate function. ISO 15998, notably, considers the failures related to communication busses involved in a safety-related function, defining a model for the transmission of safety related messages. This model presents a list of transmission errors which could cause a dangerous loss of a safety function, together with a set of countermeasures which allow to to detect such errors and enter the safe state. A summary of this model is given in table 2.1.

2.2 Hardware Architecture for Safety-Critical Systems

In this section some hardware architectures commonly used in safety-critical systems are presented. As a main reference for hardware categories, ISO 25119 is used, but the concept can be found also in other standards.

	Running Number	Time stamp	Time expiration Reception	acknowl-	Identification for sender and receiver	Data integrity assurance	Redundancy with cross-check	Different data integrity assurance for SR and non-SR messages
Repetition	X	X					X	
Loss	X			X			X	
Insertion	X			X	X		X	
Incorrect Sequence	X	X					X	
Message Falsification				X		X	X	
Retardation		X	X					
Coupling of SR and non-SR information				X	X			X

Table 2.1: Efficiency of measures in case of possible transmission errors, redrawn from ISO 15998:2008, Annex D, Table D.1

2.2.1 ISO 25119 Hardware Categories

There are four possible hardware categories as shown in Figure 2.2. In each architecture there are some common blocks and links:

- I input block
- L logic block, e.g. a microcontroller, often called Main Micro Controller (MMC)
- O output block
- TE test equipment, e.g. a component that tests the MMC for errors, often called Safety Micro Controller (SMC)
- OTE output of test equipment
- S_I interconnection with input signal
- S_O interconnection with output signal
- m monitoring
- c cross-monitoring

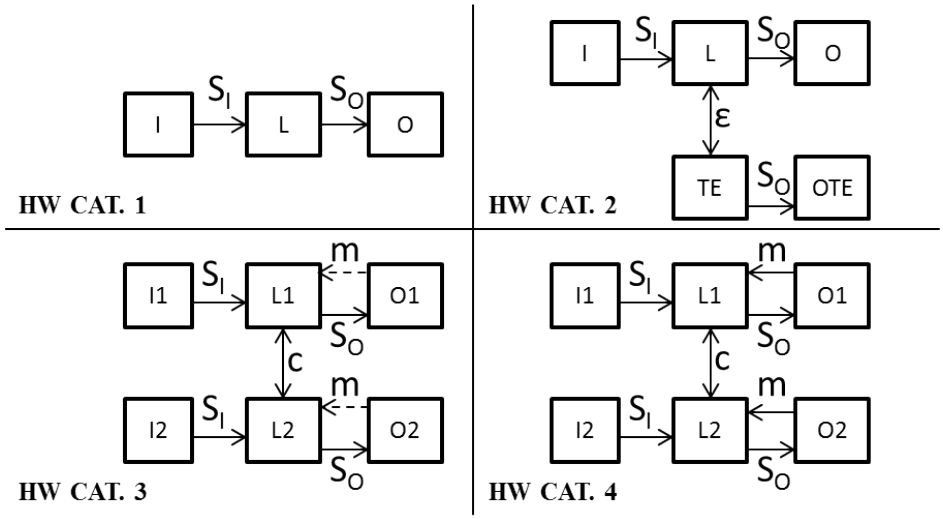


Figure 2.2: Hardware categories according to ISO 25119.

The first one (Category 1) has a single and unidirectional chain from input (I) to output (O) passing through the logic (L), which generally corresponds to a microcontroller. This is referred to as main microcontroller (MMC). No diagnosis on input or output by the logic is required. If a similar hardware structure is used in a safety critical application, it is required to use only well tried devices and components. The second one (Category 2) adds a test equipment (TE) that usually corresponds to a second microcontroller, which has the role to monitor the logic L and, in case of fault occurrence, it must act on the output to reach a safe state. This is referred to as the safety microcontroller (SMC). In this category a fault either in the input or in the output could be diagnosed by the logic, increasing the overall Performance Level (PL) from the functional safety point of view. The third one (Category 3) has a complete redounded chain, composed by the input blocks I1 and I2, the logic blocks L1 and L2, and the output blocks O1 and O2. The logic blocks cross-monitor each other. In case of faults it is possible to continue working, if the faulty channel (I1-L1-O1 or I2-L2-O2) is turned off by the other one. In this case the system is fault-tolerant. The fourth hardware category (Category 4) has the same architecture of the third one but with a higher diagnostic coverage requested on the output (continuous instead of discontinuous tests).

2.3 Software Quality

From a normative point of view the quality of the software consist of a set of measures to reduce the probability of a dangerous systematic error, given by an incorrect implementation of the safety specifications. The whole software life cycle must be considered, in parallel with the product life cycle, from the specifications to the implementation and testing.

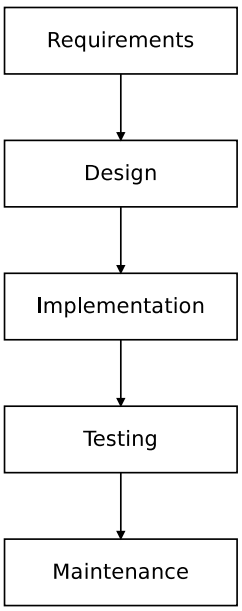


Figure 2.3: The Waterfall software development cycle.

2.3.1 Software Development Cycle

The development cycle of software is composed by a list of subsequent phases and a certain temporal execution. Common development phases include specification, implementation and testing.

Waterfall Development Model

In the Waterfall development model, the requirements of the software are considered fixed and immutable, and there is a strict temporal succession of specification, design, implementation, testing and maintenance phases. This is the oldest software development cycle and has the disadvantage of imposing a very high cost to any modification to the specification, design or implementation if a deficiency is discovered at later phases, for example during testing.

V-cycle Model

The V-cycle software development model, like the waterfall model, defines a specification, design, implementation and testing phase, but it allows for some interaction between different phases. It clearly defines how a change should be handled when required for example during the test phase, if a problem is discovered. With reference to Figure 2.4, a generic V-cycle represent the execution phases with reference to the timing, like the Waterfall model, and it pictures the phases at different levels depending on the level of abstraction, from a high-level functional specification to a low-level implementation detail. Direct feedback is allowed only between phases at the same level of abstraction.

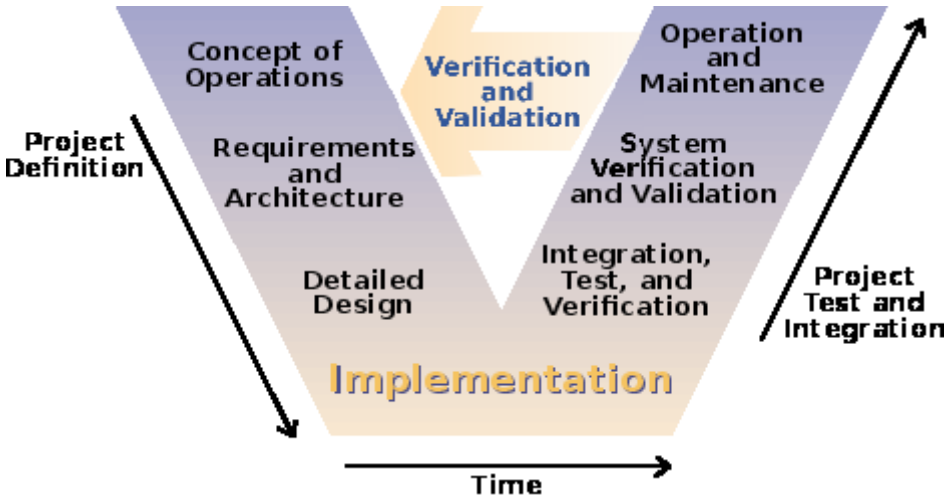


Figure 2.4: The V-Cycle software development cycle.

2.3.2 Software Specifications

Functional safety standards demand that the functional safety requirements of the software must be traced through the whole product life-cycle. Many commercial tools exists for this purpose, for example IBM Rational[2] and Polarion[3]. The manner in which software requirements are expressed depends on the required integrity level. For lower safety integrity levels, software specifications in natural language are considered sufficient; increasing the required integrity level there is a need to specify such requirements either with semi-formal methods (flow charts, diagrams and figures) and formal methods (finite state automata, mathematical models, analysis and notation).

2.3.3 Software Implementation

ISO 25119 contains a detailed description of implementation techniques for safety critical software, depending on the required SRL. In particular there is a requirement for strongly typed languages; this clearly means that the standard C language is not suitable for safety critical systems, so one solution is to use at least a language subset[4] which enforces a stronger typing than the standard. One well known subset is MISRA C [6], which is widely used in the automotive world. One problem of using such subset is that it is not trivial to enforce it, especially if legacy code is to be reused, because it forbids many constructs of the language which are commonly adopted, especially if the code was written using older revisions of the C standard like C90. For this reason it is mandatory to use one or more automatic analysis tools to verify the compliance to MISRA C. An alternative would be to use languages with a native

[2] https://www-01.ibm.com/software/it/rational/

[3] https://polarion.plm.automation.siemens.com/

[4] A language subset is actually a separate requirement. However using MISRA C can fulfil both requirements, being itself a subset of C and forcing a stronger typing model than plain C.

strong type checking such as Ada, but it is usually considered to have higher adoption barriers, and for this reason it is not widely used. Another solution is to use higher level languages and a model-based approach (e.g. using Mathworks Simulink or Design Verifier), where the application logic is described in a way much more abstracted from the hardware and much closer to a mathematical model. Even in this case it is still useful to use some automatic tool or language subset to avoid common errors. Aside from the programming language, some additional requirements are related to the cyclomatic complexity of functions, the use of coding standards, avoid the dynamic allocation of resources.

2.3.4 Software Testing

Software testing, in the field of safety critical systems, aim at verifying that the implementation corresponds to the software safety specifications. With reference to the V-cycle model in Figure 2.1, testing must be performed both at the module level and at the integration level, and finally at system level. Depending on how the specification are redacted, different testing methods are possible in order to provide an appropriate guarantee that the implementation correspond to the safety specifications. The most common testing method is to create test cases or unit tests, that is the program is executed to perform a very specific function and the output result is compared to the expected result. This is also used to perform regression tests, that is after a modification the tests are repeated to ensure that no functionality is lost. Other test methodologies are Model-in-the-Loop (MIL), Software-in-the-loop (SIL), Hardware-in-the-Loop (HIL).

2.4 Safe Communication

Recent standards like ISO 25119 in Annex B, and previously ISO 15998 in Annex D, consider the composition of single ECUs in a network, by requiring some techniques to be used in communication systems to detect communication errors, which could otherwise result in a dangerous condition of the system. While ISO 15998 is more concerned to the typical transmission errors, as described in section 2.1.4, ISO 25119 is more focused on the partitioning of different safety functions over a communication network. Additionally, ISO 25119 in Annex B considers the requirement for software partitioning in a single ECUs, with the purpose to have software functions of different criticality in the same ECU. In this case, the separations between different execution domains must be guaranteed for the memory and time domain as well as inter-task communication.

Chapter 3

Vehicular Networks: an Evolutionary Perspective

In this chapter an overview of current vehicular network technologies is given. Starting with an introduction of current and future wired technologies, then some examples of wireless applications are given. An overview of security issues and how they are currently handled in the agricultural and heavy-duty field is then given. This is motivated with the fact that the recent introduction of connectivity, linking the wired vehicular network with Internet or some external device, using one or more wireless technology, has the inevitable consequence of exposing vehicular networks to security threats. Finally the In-Tractor cloud paradigm is exposed, which represent the future vision of vehicular network, developed by the author together with the research group at CNR-IMAMOTER and ESTE S.r.l.

3.1 Wired Networks overview

Since the introduction of vehicular networks, in order to connect two or more Electronic Control Units (ECUs), the main communication technology has been the *Controlled Area Network bus* (CAN). It was invented in the years '80, almost thirty years ago, was standardised in 1992 with version CAN 2.0 and finally became an international standard, namely ISO 11898, which last revision is dated 2015 [7]. The CAN bus uses a base-band communication with bit-stuffing, allowing a maximum of 8 byte of payload and an identifier of the message which can be 11 or 29 bit long. The channel access is controlled by the identifier using a Carrier Sense Multiple Access with Bit Arbitration (CSMA-BA) method, where the identifier represent the priority of the CAN message and exploits the bus nature of CAN to perform an arbitration between colliding messages. Standard raw bit rates are 250 kbit/s, 500 kbit/s and 1000 kbit/s. While the CAN bus is still today the main and most widely used vehicular network, thanks to the very low cost, many other technologies have been used in small sub-network, like

Local Interconnect Network (LIN) for sensors and actuators. The increase of complexity in electronic control applications, for example the introduction of ABS in cars and then x-by-wire applications, required real-time capabilities which are not available on standard CAN networks. For this reason Flexray was introduced, using a TDMA access method instead of the classic CSMA-BA and allowing up to 256-byte packets at 10 Mbit/s.

Multimedia applications, like audio and video streaming, are now the key enabler for high-speed vehicular networks. One of the prominent technologies is Ethernet, enabled by the IEEE 802.3bw BroadR-Reach automotive-compliant physical layer.

In the agricultural field, the standard protocol stack is called ISOBUS, standardised as ISO 11783 [8] and based on the CAN bus. Previous research work at CNR-IMAMOTER was aimed at developing an *ISOBUS 2.0*, also called high-speed ISOBUS, based on Ethernet and suitable for control applications. This requires at least soft real-time capabilities, which is now available to some degree on commercial components and belonging to the Audio-Video Bridging standards group (AVB, e.g. IEEE 802.1AS and 802.1BA), more recently generalised as Time Sensitive Networking (TSN e.g. IEEE 802.1Qat and 802.1Qcc for stream reservation).

3.2 Wireless Network Overview

In the last years wireless technology appeared as a key component for traditional and new applications in agricultural machines. Common applications include:

- **Fleet-management** Usually the infrastructure is composed of remote wireless connectivity, like 3G and more recently 4G. It allows for example real-time positioning of the machine fleet, download of the field map, upload of operational data. While the data format is standardised as ISO-XML (also called AGRO-XML), the protocols used for the remote communication are usually proprietary.

- **On-field and on-machine sensor networks** This is a growing application field, with application in precision agriculture, field monitoring[1], load/weight sensing, Tire-Pressure Monitoring Systems (TPMS)[2]. Other applications are in cooperation with drones, which fly over the field to gather the data transmitted by short-range low-power sensors.

- **Human-to-machine interaction** This is still in its early stages, but some implements start to have short-range connectivity like Wi-Fi and Bluetooth, which can be exploited by the operator for example through an application on a mobile device.

[1] http://www.orange-business.com/en/dacom

[2] http://www.stecom.com/Case_history/Stecom_Case_history_Tire_Pressure_Monitoring_System.html

- **Machine-to-machine communication** Wi-Fi connections, based on IEEE 802.11 standards, are used for M2M (machine-to-machine) communications. A popular example is an harvester machine which cooperates with a storage truck. They are connected through a mesh network and exchange real-time data with the position and other operational parameters, so the storage truck is capable of following the harvester machine at the right distance and with the adequate speed.

There are two main challenges that vehicular networks need to address currently: *security* and a *sustainable architecture evolution*. The first challenge arises mainly from the use of wireless connections, which are inherently more vulnerable to a malicious operation than wired network. In fact, historically vehicular networks have always been considered as trusted network, with a strong commitment to safety. The second challenge derives from the consideration that new applications often required a drastic change in the underlying network architecture, as well as the architecture of other applications. Since this is a very expensive process, the challenge is to find an architecture which can survive this evolution, in part or completely.
Wireless networks are an enabler technology for the interesting applications that are currently developed (although not the only one), so they concur in exposing both security issues and the need for architectural changes. These two aspects are treated in more detail in the following sections.

3.3 Security Issues

Cars and heavy-duty machines are not anymore just isolated entities, but nowadays they can be considered devices, like computers, tablets and smartphones, connected to the Internet. This great opportunity gives to cars and heavy-duty machines new important functionality that makes them more appealing. However, the Internet connectivity make vehicles exposed to cyber-security threats; for instance, malicious users may remotely find and exploit some vulnerability to remotely access the car and perform malicious actions. A well-known attack is that on the Jeep Cherokee performed by Valasek and Miller [9] and published in the summer of 2015. The authors of this attack demonstrated a hack to remotely control a Jeep Cherokee through the Internet connection, exploiting various vulnerabilities in the infotainment system. Basically, Valasek and Miller were able to remotely change the direction of the steering and also to command the breaking system. As a consequence, Fiat Chrysler was forced to provide a software update [10] to be installed through the USB port on their vehicle's dashboard.
Many examples of the attack possibilities and their feasibility are now present in literature; for example, in [11] the authors exploit local access like the CAN network to gain complete control of a car, while in [12] the same authors explore the possibilities of a remote attack,

always aiming at obtaining full control of the vehicle. These examples, although focused on the automotive field, are still relevant for heavy-duty vehicles, since the increased connectivity that is permeating also this field exposes to similar scenarios, although with different consequence, given the different nature of the vehicles.

Seen the above security risks, the emerging opinion is that past communications protocols, and new deployed ones, should be adapted to cover some security requirements and to avoid, or at least mitigate, the security threats. Old protocols may be extended, if possible, to embed security solutions, while new ones should be designed with the Security by Design principles, in which engineers should have in mind the attack surfaces [13] to provide the right security solutions.

In this thesis the focus is mainly on agricultural and heavy-duty machines that are equipped with one or more CAN-based networks, used by all the electronic control units (ECUs) to communicate. In the last years, due to the increasing demand for bandwidth from the applications implemented in the ECUs, the Ethernet network has been considered as a long-term replacement for the vehicle networks in agricultural and heavy-duty machines. On the other hand, there is also an increasing need for secure communication, mainly due to the fact that in-vehicle networks are now connected directly or indirectly with other networks, and in many cases with the Internet. This increase in connectivity opens new technological and business possibilities but, as reported in [14], it creates a situation where a vulnerability can have disastrous consequences, both in terms of security and safety. The effective implementation of these two scenarios depends to a large degree on the capabilities of the underlying network; as shown in this section, the traditional CAN networks ultimately impose some limits on the security level that can be obtained, mainly due to the very short maximum packet size. On the other hand, an Ethernet network permits a greater packet sizes, and the security level achievable is basically given by the computational power available on the involved ECU.

CAN-based networks are here analysed considering three different security properties, which are: *Authentication, Integrity* and *Confidentiality*. In particular, some relevant and well-known security threats are exposed, investigating how sound the CAN networks are against the chosen security threats. Moreover, some security solutions are proposed, when possible, that could make the current CAN standard more robust. A comparison of the security solutions and threats in case the CAN bus is replaced with the Ethernet network concludes this section.

3.3.1 General Security Considerations

In this section the following security aspects are introduced: *attacker model*, *security threats* and *challenges*.

Attacker Model

Nowadays, the evolution of communication technology into heavy-duty machines can introduce new vulnerabilities that impact on the proper machine functionality. Section 3.3.1 lists some security threats that attackers could exploit being located far away from the vehicle, or in proximity, using a physical connection with the vehicle, e.g., a USB-port. The choice to consider versatile attackers is enforced by the fact that vehicles are not isolated entities anymore, but they are connected to the Internet through, for instance, 3G/4G or Wi-Fi connectivity. Thus, by exploiting wrong security configuration or security flaws in the protocols, attackers may easily gain access to the machine without particular effort. In fact, if we consider that a generic communication protocol, developed in the past and used for decades, was not thought to be secure by design, for example without an authentication system, once it is moved into a different domain, e.g., the Internet, attackers can access the vehicle's local network without any hindrance.

Security Threats

The lack of proper protection for ECUs inside heavy-duty machines makes them exposed to potential attacks. In particular, the risk of attack increases when vehicles are connected to the Internet to provide additional services. An attacker may exploit remote connectivity to get access to the machines and execute malicious actions. However, remote attacks usually require considerable effort to be exploited. For instance, if ECUs are configured to request basic authentication, i.e., username and password authentication, to access their internal services, then the attacker must be able to break the authentication system to get partial or full control of the vehicle itself and execute several actions, e.g., commanding the vehicle trajectories, sniffing messages of the ECUs and so on.

Here some attacks are listed, well-known into the security field, that can bring relevant security issues to heavy-duty machines. Also, given the nature of the attacks, it is useful to classify them into two categories: *Local* and *Remote*. In the first category, an attack is executed only if the attacker is placed in proximity of the target; in the second, an attacker exploits the Internet connection to access the vehicle.

Starting with the *Sniffing* attack (*Local*) a malicious user may be able to read the content of all messages exchanged through the CAN bus. In fact, as soon as messages do not contain the confidentiality property[3], any message is vulnerable to this attack.

An attack linked to the sniffing, and in the same local category, is the *Replay* attack. This typology of attack is bounded to the first one since it uses messages captured with the sniffing

[3]Generically speaking, for *confidentiality* is meant the possibility to exchange a message between two users without that a third user is able to read the content.

attack that are retransmitted on the network for specific purposes. For example, an attacker may sniff and store communications between the ECU that controls the wheel, and the ECU that controls the wheel movements, to replay the actions without control of the driver. Once the attacker captures the command to perform the given action, it can send again the command to the ECU to generate the same movement of the wheels.

Moving to the remote attacks, the first described is the *Denial of Service* (DoS) attack. Here, the attacker wants to make unavailable one or more services of the vehicle. This kind of attack is common on the traditional communication networks, and in particular it involves attacks on Web Servers and Web Services, to make them not reachable from end-users. Within heavy-duty machines an attacker, who performs the DoS attack, can make specific ECUs unavailable , e.g. targeting the ECU that controls the wheel steering, the DoS attack strongly impacts the safety of the vehicle.

Another remote attack is the *Black hole* [15], which can be considered a special case of DoS attacks. In this case, the attacker remotely gets control of one or more ECUs to block and drop messages that transit through them.

Security Challenges

As presented in section 3.3.1, heavy-duty machines may be exposed to threats that could impact their security and safety. This risk increases when vehicles are connected to the Internet and they become remotely reachable, for instance exploiting their public IP address. So, vehicles cannot be seen as isolate entities protected from computer security attacks anymore. However, heavy-duty machines connected to the Internet should be protected from attacks listed above, but presenting solutions already available “in the market" is not always feasible due to the current physical infrastructure available in heavy-duty machines. For instance, confidentiality may be considered by importing cryptographic solutions to encrypt messages. Another solution that may mitigate the risk of DoS attacks could be using firewalls at the border of the vehicles' network, to block message flooding that may make services unavailable. Also, message validation techniques may help decrease the impact of a DoS attack to ECUs by sharing a secret that they will use to evaluate the validity of messages, that is to authenticate it. So, an attacker must be able to forge a valid message to be accepted by the counterpart, and this is not trivial since it requires that the attacker guesses the secret, e.g., an attacker may execute the *Random message* attack to forge random messages.

The above solutions can help the heavy-duty machines to “raise the bar" on their level of protection against security attacks. However, those solutions require a suitable network infrastructure and enough computational resources to be implemented. The goal here is to analyse the current network infrastructure available in the heavy-duty machines, i.e., CAN bus,

and to propose solutions that reduce the risk of attacks presented in 3.3.1. Finally, the same solutions are compared with those possible on an Ethernet network, analysing their feasibility.

3.3.2 General Network Considerations

In the following sections, the requirements of typical periodic real-time traffic are considered. For a security measure or algorithm to be applicable, the following constraints must be fulfilled:

1. **Limited computational complexity** the execution time of the algorithm must be negligible compared with the time repetition of the messages. Ideally this should also be fixed to avoid timing attacks and satisfy real-time requirements;

2. **Minimum overhead on the data packets** the algorithm overhead in terms of bytes added to the message must be negligible with respect to the network packet constraints, in order to be transmitted without fragmentation. This is required for many real-time messages, where fragmentation is often not tolerable.

While the computational power available on a typical ECU can be expected to increase over time, due to technological advances, the network packet constraints are generally fixed and depending on the network technology used. Furthermore, the network usually also imposes some requirements on the possible topology and overall achievable security level.

3.3.3 Present - CAN network

The CAN bus is one of the most widespread vehicular networks and is used on automotive, as well as heavy-duty, agriculture vehicles. For a plain CAN network the maximum message size is *8 bytes*; although various Transport Protocols like ISO-TP [16], SAEJ1939 Connection Mode [17] or Extended Transport Protocol [18] are available, which ensure larger messages to be transferred, but they impose an additional overhead that makes them typically not suitable for real-time traffic, since they split a single block of data over different messages and use explicit acknowledge packets for reliability. A CAN network imposes a network architecture that is a physical shared bus, therefore everyone can listen to the messages generated from other ECUs and can also transmit messages which can be received by any ECU. This exposes to *Replay attacks* and data collection (*Sniffing attack*) for offline attacks on the algorithms used, as well as *DoS* and *Black Hole* attacks.
A typical vehicular network is composed of more than one CAN subnetwork, usually identified by the main purpose of the connected ECUs, e.g. the powertrain network (e.g. transmission, engine control), the implement (e.g. ISOBUS) network and the diagnostic network. A single ECU often has more than one CAN interface, and can be connected to more than one CAN

subnetwork. To mitigate these attacks, CAN gateways could be employed, but in practice they are rarely used.

The different CAN subnetwork can also have different exposures; usually the powertrain network is a private internal network, while the ISOBUS network is open to the connection of third-party implements, which are often out of the control of the manufacturer. Similarly, there is usually a diagnostic network, which can be separated from the others but can also be part of an existing CAN subnetwork. It must be noted that currently multiple networks are generally adopted not to increase security, but because a single network is not sufficient for the throughput requirements of the applications, or because of some dedicated sensor network.

Authentication over CAN bus

Currently, on a CAN network, authentication is only performed to enable certain functionality, typically for maintenance purposes. The protocol used is generally called *Seed and Key*, and is used for example in the CAN Calibration Protocol (CCP) [19] and in the Keyword 2000 protocol (KW2000) [20]. This protocol is performed in two phases:

1. the client ECU that wishes to authenticate on a server ECU requests a seed S, usually a random number. The client ECU then computes the key K, according to a certain function f, depending on S and optionally on a fixed key k_s shared between the client and the server;

2. the server ECU receives K from the client ECU, and verifies its correctness in order to decide if the functionality has to be enabled, usually by recalculating it with the same algorithm used by the client ECU.

Neither the CCP nor the KW2000 protocols specify the algorithm to be used for the *Seed and key*; such algorithm can depend on the application, the manufacturer and the functionality. These algorithms usually rely on security through obscurity (for example $K = f(S, k_s)$ where f is a secret function, which sometimes is a simple addition, or a bit-wise xor). It is worth noting that the main purpose of the *Seed and key* protocol is to reduce the chance of unauthorised operations; however sometimes the secret algorithm is discovered, by means of reverse engineering, for example for tuning or cloning of ECUs. The *Seed and key* protocol is usually vulnerable to the *Replay attack*, since for a given seed S_i there is only one possible key K_i. Although the manufacturer of an ECU can limit the number of *Seed and key* transactions per second to mitigate the possibility of a *Replay attack*, the number of S_i, K_i pairs that can be obtained in reasonable time is normally sufficient for reverse engineering the secret algorithm. The *Seed and key* protocol as currently used is clearly not sufficient for the security goals exposed here, since the data must be continuously authenticated, and the authentication must

be difficult to break. This means that every CAN message should be authenticated, using a Message Authentication Code (MAC). The authentication of a CAN message, whose size is limited to 8 bytes, can be problematic since it requires an additional field to be added to the message. Using multiple messages could not be a feasible solution for real-time traffic, as noted in section 3.3.2. While it could be possible to limit the data message length reserving a number of bits for a MAC, another solution could be to use the existing 15 bits normally used for the CRC field to store a 15-bit MAC code, therefore substituting to the integrity check, or to allow the use of DLC in the range 9-15. This has both the disadvantage of a limited MAC length and non-standard behaviour, but allows for full 64 bit messages to be authenticated. However the need for a custom transceiver is likely to be a major hindrance for these extension of the CAN protocol. A similar approach with a modified CAN bus is presented in [21].

Given the size constraint, the strength of a MAC depends not only on the algorithm used, but also on the length of the MAC itself; in a standard CAN message, this will be the number of unused bytes. For example, using the CMAC scheme, an evaluation of the minimum MAC length is given in Annex A of [22]:

$$Tlen \geq log_2 \left(\frac{MaxInvalids}{Risk} \right)$$

where $Risk$ is the highest acceptable probability that an invalid message is recognised as valid and $MaxInvalids$ is the number of maximum invalid messages before the key used to authenticate the message is retired. For example, a MAC of 1 byte is sufficient to guarantee a probability of 2^{-7} if the key is retired after 2 invalid messages are detected. An analysis of the bits required to tolerate a certain number of invalid messages and the number of old packets to keep in memory is given in [23]; here the authors extend the idea to different MACs in the same message, making it suitable for multicast authentication. The limitation of message size could be addressed using a transport protocol, but this will result in an increase of the busload, in addition to a greater latency given by the reassembly of the data.

In existing standards like SAE J1939 [17] or ISOBUS [18], the majority of messages do not have room for a MAC, since all the 64 bits of CAN are used. In this kind of messages, proper authentication is considered not possible using the standard CAN, unless additional messages are allowed.

Authentication schemes such as Leia [24] provide lightweight authentication and require the use of only one additional CAN message for each message to be authenticated. While this could be acceptable for highly critical messages, its use of the CAN ID is not compatible with ISOBUS. The importance of using the CAN ID here lies in the transmission of a counter related to the authentication protocol and to operation like synchronisation between ECUs.

Other authentication mechanisms have been proposed for extensions of the classic CAN bus like Flexray and CAN-FD, see for example [25] for a comparison.

Integrity over CAN bus

Message integrity is normally achieved on a CAN network with a CRC16 checksum. While CRC16 is not suited for cryptographic purposes, since for example it is a linear function and it is easily invertible, it is not easy to modify a message in transit without being detected, because of the bus nature of the CAN network. It is however feasible, as demonstrated by the CaCAN scheme [26]. A replay attack, however, is feasible if messages are not authenticated, since an attacker can inject packets in the network pretending to be anyone, also with higher priority than legitimate packets. Additional checksum or hash algorithms can be used; however in some standard messages (e.g. SAEJ1939 TSC1 - Torque Speed Control 1) there is no room for additional integrity checks. The possibility to increase the integrity protection of the message depends then on the specific data and the admissible data ranges, which can be given by the protocol (e.g. command codes) or physical meaning (coherency control).

Confidentiality over CAN bus

The most common way to achieve confidentiality on a communication is using some form of encryption. Given the security requirements listed in section 3.3.2, a natural choice for CAN bus is a symmetric block cipher, which block length is exactly 64 bits; algorithms like DES, Blowfish are then well suited for this purpose. A schema of the encryption and decryption procedure is visible in Figure 3.1. Generally, for short term security a symmetric key of 96 bit is sufficient, raising to 112 bit for middle-term security and 128 bit for long-term security, as reported in Table 3.1. However, for real-time traffic, a 64 bit symmetric key size might be enough, if a key refresh procedure is considered. Another interesting class of algorithms to consider is the class of stream ciphers, which allow the encryption of exactly the number of bytes needed. However, this needs further considerations, since this class of algorithms is often weak against sniffing attacks. This can be mitigated adding a time-variant component to a message, for example xor-ing the plaintext with a time-varying data called nonce. A greater security level, intended as an increased key size, can be obtained using a symmetric block cipher in Counter mode (CTR) mode, as shown in Figure 3.2. Here the plaintext data are not directly encrypted, but they are xor-ed with the result of encrypting a nonce. However this come to a price, that is the involved ECUs must have a shared knowledge of a time-variant value, such as a running number, which must be used to construct a nonce that must be different for each encryption in order to guarantee the security of CTR mode. A distributed running counter such as the one used in [27] can be used. The key exchange procedure will

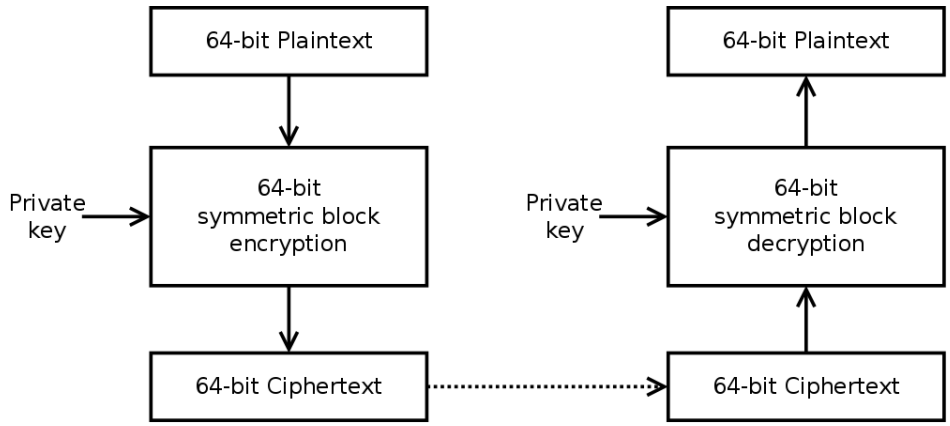
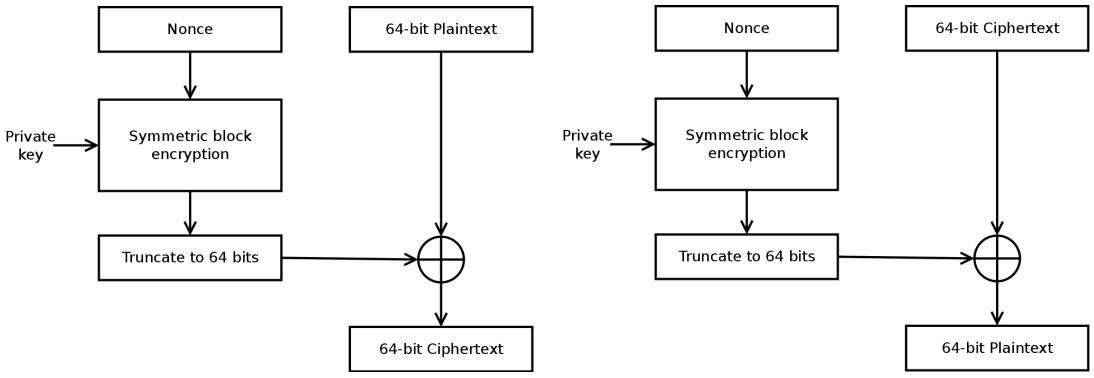


Figure 3.1: Encryption and decryption procedure.



(a) Encryption with CTR mode.

(b) Decryption with CTR mode.

Figure 3.2: Encryption (a) and decryption (b) procedure with CTR mode.

need a transport protocol in this case; in order to achieve a sufficiently fast key exchange and computation, it is essential to use short key sizes. For this reason, the best algorithm to use for key exchange is an Elliptic Curve algorithm with a key size that will be a compromise between the security grade and the overhead (bus and computational) required.

3.3.4 Future - High-speed network (Ethernet)

Modern Ethernet network are designed to permit a greater packet size, up to *1500 bytes*, if there are no additional tags such as 802.1q or 802.1ac. Furthermore the topology can be both a physical bus (using hubs to interconnect the ECUs) and a segmented network (using switches to separate the collision domains); the possibility of using a switch or router is essential to mitigate *Replay*, *Sniffing DoS* and *Black Hole* attacks, since they can filter the traffic only to the links directed to the source and destination ECUs; furthermore, it is possible to create different VLANs according to IEEE 802.1q [29] to logically separate different network segments, which can physically share the same link.

Security Level	Symmetric	ECC	RSA
Very short term, weak attacker model	64	128	816
Short-term (Standard level)	96	192	1776
Middle-term	112	224	2432
Long-term	128	256	3248

Table 3.1: Recommended key length in bits for asymmetric-key and symmetric-key cryptography [28]. Additional references are available at http://www.keylength.com.

Authentication over Ethernet

In Ethernet-based networks, authentication can be performed at different levels. Here the packet length can be much greater than CAN networks, so for single messages a MAC code can be used without causing fragmentation. The use of an asymmetric key to authenticate the messages, for example with a digital signature scheme, is not a viable solution, given the requirements listed in section 3.3.1. Although asymmetric cryptography, in particular ECC, has been shown to be feasible on constrained embedded platforms, for example in [30], the encryption and decryption time still is not sufficiently low for real-time traffic, where the maximum time allowed for generating an authentication token can be the order of milliseconds. For this reason, the best approach is to use a Message Authentication Code (MAC), which requires a key exchange procedure. One example of such algorithms is HMAC [31]. Asymmetric key cryptography is however still valid if used as part of the key exchange and key refresh procedures, since in this case it is admissible to have a low-priority task which takes hundreds of milliseconds to terminate.

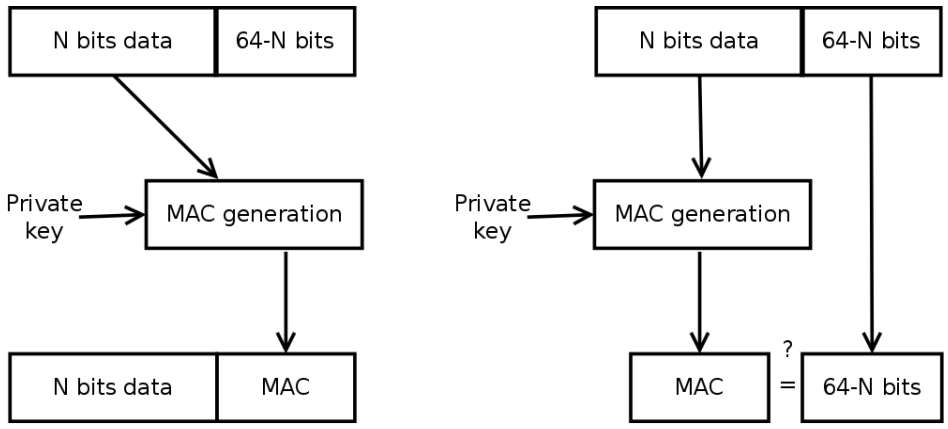


Figure 3.3: Authentication and Verification procedure with a MAC code.

Integrity over Ethernet

On an Ethernet network, data integrity is only assured at most on a link basis with a 32-bit CRC; this is due to segmentation. While one could argue that a switch could not modify the CRC, in practice there is no such guarantee, and a switch could also regenerate the CRC for all egress packets; in this case, an error during the transit of the packet from the ingress queue to the egress queue would cause an integrity loss. In other words, this is only a hop-to-hop integrity check. End-to-end integrity is usually provided by higher layer protocols, for example by UDP and TCP [32]. In general, Ethernet allows for higher integrity levels to be reached, since it poses no limitation on the integrity check to use, and there are many known protocols used on IP-based networks, based on hash algorithms, which can be used for this purpose. However, since most protocols use CRC, it is not sufficient to protect from deliberate modification of a message; in this case, authentication is necessary.

Confidentiality over Ethernet

The maximum payload contained in a single Ethernet frame is of 1500 bytes, so a standard symmetric cipher like AES can be used. Moreover, the key exchange procedure does not need a transport protocol since, if sufficiently short key sizes are used, for example using an Elliptic Curve algorithm, the messages to be exchanged are easily contained in a single packet.

3.3.5 Discussion

As analysed so far, the opportunities that attackers have in the widely used CAN-networks are not negligible. The fault does not totally belongs to the CAN design, but its weaknesses come from the fact that the CAN bus was never thought to be applied in a communication domain such as the Internet, where security is paramount. Thus, an attacker, who acts as depicted in section 3.3.1, can achieve his/her goals with no particular effort since most of the basic security properties are missing. In fact, on a CAN network, due to the limited message length, a proper secure authentication method is not feasible; if a full 64-bit standard message is required and fragmentation is not allowed, an authentication method is not possible at all. On the other hand, on Ethernet it is possible to securely authenticate a message, at the cost of a relative little overhead given by the addition of the MAC. Again, the security of the MAC is limited by the computational power available on an ECU, but since most microcontrollers embed a hardware implementation of a symmetric block cipher, which can be used in CBC-MAC or CMAC mode, a secure authentication is immediately possible. If no hardware cryptographic module is present, a software implementation of such symmetric block cipher can be used; some comparison between the time required for different algorithms is presented in [33], [34]

and [35].

3.4 The In-Tractor Cloud vision

In recent years, the automotive industry seemed to be lured by the need of increased bandwidth, running after the sensible growing number and computation power of the Electronic Control Units (ECUs), especially for Infotainment services (e.g. Genivi).

For what concerns agricultural (AG) mobile compartment, new use cases and the enhancement of some existing others require the adoption of a high-speed network like Ethernet; some efforts have been already put in adopting Automotive Ethernet in AG mobile networks. These are implemented at standardisation level in ISO TC 23/ SC 19/WG1 where a Task Force has been created for the investigation of the applicability of such new physical layer. However, modern agricultural machinery, special vehicles and similarly Commercial vehicles implement a huge variety of applications, with specific controls and functionality. Thus, these sort of vehicles have networks often open to instrumentation and implements, made by third party manufactures. These points suggests a different approach in terms of both service design and operation. In the IT world the "Anything as a Service" paradigm drives the current technology, and is considered one of the most representative expression of the cloud computing [36]. By adopting this paradigm in the Commercial vehicles field, this allows to express the critical properties and requirements of the architecture, and the applications and functionality that make use of the in-vehicle network, into new concepts like for example Redundancy-as-a-Service, Synchronization-as-a-Service, Connectivity-as-a-Service; existing concepts of Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) can also be reused to realise what is here called an *In-Tractor Cloud*. A new perspective in the design and implementation of the control systems is therefore possible. On one hand, it lays the basis for advanced autonomous driving systems; on the other it offers increased scalability, reliability and, most importantly, a future-proof paradigm, effectively decoupling the design of the control systems from the physical implementation of the network and in general from the hardware, in a manner similar to a Software Defined Network. This decoupling has a profound impact on design, development and maintenance of vehicular AG networks. At present day, it is not uncommon to have 4 or 5 different physically separated networks on a single vehicle, as pointed out in section 3.3.3. Ultimately, this physical architecture is tightly coupled with the applications implemented on the In-Tractor network, and this result in the need to specialise both ECU development and physical network architecture, reducing the possibility of component reuse and network structure evolution in new tractors and specialised big agricultural Machines. This approach is far from being scalable, and the requirements of the newest applications are pushing the

network complexity to its limit of practical and economic convenience. It clearly follows that having a different, more scalable, flexible and cost-effective approach is necessary, eventually taking inspiration from the network design principles in other application fields.

3.4.1 IT Technology Principles

The history of IT technology is representative of a long-term evolution of a communication network, in this case related to the Internet. The technology evolution and the paradigms that emerged for the realisation of the services that constitute the fuel propelling the Internet expansion represent important steps, for which a parallel with vehicular networks evolution can be made. At the beginning the Internet network was essentially a trusted network, comparable to the present CAN-based vehicular network, where end-to-end connectivity was the natural paradigm of communication [37]. As the network grew, it became noticeable that it was relatively easy to perform unauthorised operations, like is happening recently in the automotive field; for example, the first worm dates back to 1988 [38]. Security was then a new argument, which led to improvements and modifications to existing protocols, for example adding a security layer like SSL (Secure Socket Layer). Similarly, security is now a hot topic for vehicular networks. With the effective application of security, new services were made possible, for example e-commerce, allowing for an increase in the value of information exchanged, given that now there were tools that could be trusted. The advent of high-speed connections (ADSL, WiFi) paved the way to multimedia applications, like VoIP and audio/video streaming. Meanwhile, with the increasing complexity of network architectures, the Software Defined Network (SDN) paradigm emerged, and is now the method used for handling the reconfiguration of complex networks [39]. This is a form of virtualization of the network functionality, allowing to handle a network connection as a service which can be reconfigured according to the application needs, effectively separating the physical implementation of the network from its logical architecture. This proved to be a fundamental and ground-breaking advancement in handling network complexity, improving scalability and reducing maintenance and upgrade cost. An analogous paradigm emerged for computing platforms, and is now known as the cloud computing paradigm. Here the hardware virtualization capabilities allowed for a whole spectrum of virtualization techniques, from distributed computing to virtual private networks. Both the SDN paradigm and the cloud paradigm share a common characteristic, that is they model resources as a service, effectively decoupling the resource provider from the resource consumer. In other words, the resource provider is not anymore localised as a definite physical entity, but is identified as an abstract entity with a defined interface, which can correspond to a definite physical entity but can also be virtualized, and be dynamically moved to different physical entities during resource utilisation. Referring to the evolutionary steps outlined in this

section, the AG networks (commonly based on the CAN bus) are in the phase where high-speed networks are being deployed, and it is still thought mostly as a trusted network. The big question that has to be addressed is: since a lot of knowledge has been developed on network and service architectures in the IT field, can the upcoming AG applications benefit from this knowledge? The answer offered by the In-Tractor Cloud paradigm is that there are some design principles that can become useful also when applied to AG networks, and can bring advantages if adopted early during the design of a new application, opening new business and application opportunities. Furthermore these principles can aid the development of a sustainable and resilient architecture, or platform, for the development of future vehicular AG networks. These principles will ease for example forward compatibility of machine ECUs and networks, allowing to focus on the applications rather than the specific physical architecture details. This will solve for example the problems on incompatibility between ISOBUS different Classes and functionality. Based on some works like [36] [39] [40] [41], interesting technological principles can be found among the Service-Oriented Architecture principles (SOA), in particular:

1. abstraction of resources (e.g. Virtualization, Description);
2. loose coupling between components, defining a standard communication platform.

ISOBUS is not a service-based network, but in some cases, the "service" idea is applied. For example, the File Server, the Sequence Controller and the Task Controller expose a kind service to other ECUs, respectively to store and retrieve files and to reproduce common sequences of operations. Both auxiliary inputs and auxiliary functions also represent inputs and functions as services, which can be bound at running time. This kind of services can be thought as traditional IT services, before the cloud and SDN era, which are localised on a single entity and tied to a precise network architecture. There is no abstraction from the real implementation (except for the abstraction provided by the ISO/OSI model, to which ISOBUS does not even adhere formally), and all functionality and services are related to the actual network structure and hardware components that are present on a specific machine. This is because the resource definition is not abstracted from their physical implementation (e.g. a joystick, moving up or down an electro-hydraulic lifter). This approach leads to limits and lack of global performance control on task and machines functionality, so having observability and controllability issues left open.

In this context, the aforementioned principles are applied in the following way:

1. abstraction leads to services that are focused on the exploitation of the resources of the machines, without the hassle of strive with network details and constraints;
2. services can be loosely coupled from ECU characteristics diversities (performance, manufacturer, development approach, internal architecture). This allows for example an

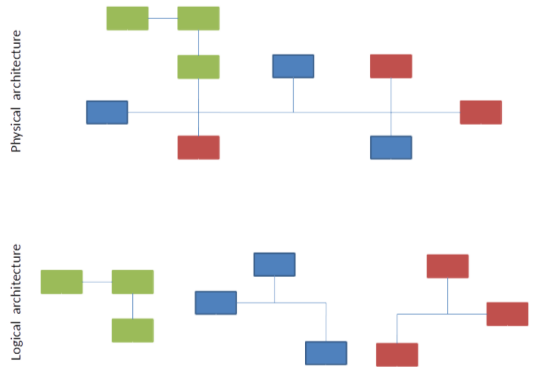
application that requires the exchange of big amounts of data to ignore the fact that other applications are using the network. In case of network saturation, the big data transfer will automatically be stopped or continue at reduced speed if tolerable, without the need of foresee in advance all possible concurrent applications, as explained in more detail in the next chapter;

The following sections will expand these principles in the context of vehicular agricultural networks, motivating their adoption and finally proposing some use cases enabled by these design principles.

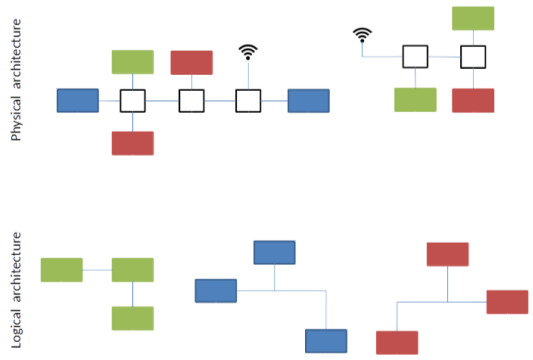
3.4.2 The In-Tractor Cloud

The In-Tractor Cloud architecture is represented in Figure 3.4(b), while a traditional vehicular network is represented in Figure 3.4(a). Figure 3.4 represent in both cases the physical and logical structure of the network, where there are three applications, divided by colour. The physical architecture is composed of ECUs, while the logical architecture is composed of logical blocks of the application (e.g. input module, computation module, storage module, logging module). In Figure 3.4(b) the logical blocks are service available on the network. In Figure 3.4(a) the number of ECUs in the logical structure equals the number of logical blocks; furthermore, the interconnection between the blocks of an application is the same. On the other hand, in Figure 3.4(b) the physical architecture can be completely different from the logical architecture. For example, the blue application has 3 logical blocks, which are mapped to 2 physical ECUs. The green and the red application, instead, transparently use physical ECUs which are connected both with a wired network and a wireless network. Finally in the physical architecture the ECUs are not directly connected, while at the logical level the logical blocks are connected. The In Tractor Cloud is an abstraction layer that hides functional, technology, programmatic logic and quality of service information ([42] from page 218). The overall scenario is visible in Figure 3.5: an In-Tractor Cloud is connected to an Implement Cloud, and the interaction is based on the concept of services.

A key characteristic of the In-Tractor Cloud is then its service-oriented nature. A service-oriented architecture is characterised by several design principles; in particular here composition and abstraction of services are considered. While other characteristics such as service interfaces and discovery are also necessary for a service-oriented-design, their applicability has yet to be thoroughly investigated, and currently represent an active field of research at CNR-IMAMOTER. A service-oriented architecture, if applied to the control network of a generic machine, can be advantageous because it would allow the re-allocation of functions. Such re-allocation can be imagined in different dimensions, *spatially* and *temporally*. A *spatial reallocation* of tasks can allow a part of the machine (e.g. a fixed ECU set on the machine)



(a) Traditional architecture of a vehicular network.



(b) In-Tractor Cloud network architecture.

Figure 3.4: Vehicular network architectures showing the relation between application and physical implementation.

to aid the operation of another component (e.g an implement), which is otherwise limited in functionality. This results in smaller and cheaper implements, if some functions (computation, logging, synchronisation) can be offloaded to the machine. Given a high-speed connection between the implement and the machine, we think that many functions, also critical functions, can be offloaded while retaining the control on its execution. The *temporal reallocation* implies the service to resize its impact on the machine and on the ongoing treatment, modifying the functional requirements (e.g the absolute speed of the machine) and/or serialising simultaneous functions. For example, the operations of an implement can be automatically adjusted, as discussed before, according to the operating conditions of the machine (speed, direction, computational load on the ECUs), even if at some point the machine is upgraded with new functionality. From the perspective of a given application, all the operating conditions of the machine can be seen as resources that are used: from those who are strictly related to it (actuators, sensors, ECUs, etc.), to the collateral ones (network infrastructures, computation time, processor cores, data memory, the code footprint, etc). These may be defined a-priori

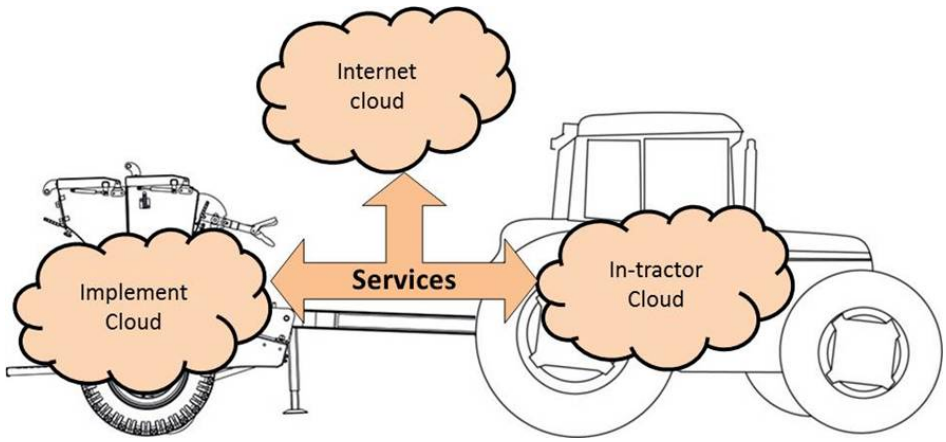


Figure 3.5: Encryption and decryption procedure.

(like currently in ISOBUS) or assessed at runtime. In particular, the upgrade of the machine can be smoothed by applying both the resource abstraction and loose coupling principles between the components. While the first ensures that the vehicular network itself can be extended, the second is of fundamental importance for the forward-compatibility of ECUs, which can then be upgraded gradually. This is in contrast with the tight coupling existing between each ECUs on a modern tractor. Such loosening could reduce the performance of the network, since it forbids cross-layer optimisation, but we think that the performance reduction will largely be compensated by the advantages in terms of components reuse and overall cost reduction. It is possible as well that the performance in some cases increase, for example because of the use of multicast protocols instead of unicast.

Resource-as-a-Service (RaaS)

The Resource as a Service paradigm emerged in the IT world, following the Anything-as-a-Service paradigm [43] and it had, and continues to have, a great impact on the economic opportunities and the way computing resources are perceived and used. The Resource as a service paradigm has been recently proposed in industrial context [44], where the authors have obtained a better resource utilisation and therefore an improved efficiency, ultimately increasing the performance of a given control network. Here, the resources treated as services are limited to communication and data resources, excluding infrastructure and platform resources. For vehicular networks the RaaS paradigm means that the operations of performing a computation (e.g. to update a setpoint of an output, computing the output of a real-time control loop, filtering of some input), having a communication channel with a given QoS (for real-time and/or reliable tasks), storing and retrieving data (e.g. logging) must be abstracted, using a defined interface instead of directly accessing the resources; adopting the RaaS paradigm allows to build a common platform for all the applications, be them control application, monitoring,

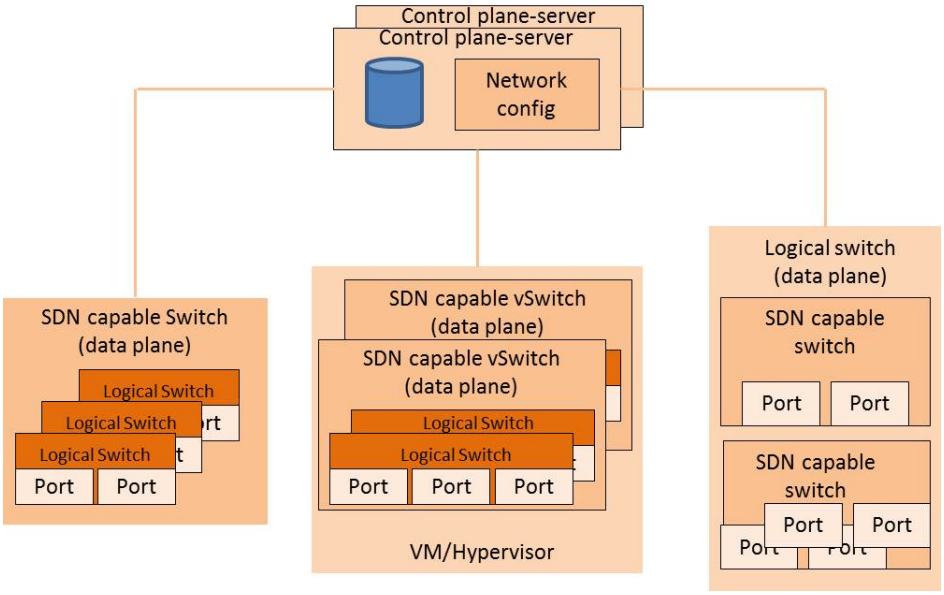


Figure 3.6: Implementation of the Resource-as-a-Service paradigm (RaaS) by means of a Software Defined Network (SDN)

auxiliary, diagnostics, or maintenance applications. An example of implementation of the RaaS paradigm by means of a Software Defined Network is reported in Figure 3.6. In order to uniquely identifying the actors a namespace is necessary, in order to organise, identify, reach, and refer to the resources. Taking another example from the IT world, the Amazon Resource Name[4] is the mean to encode all the information needed to use a given resource. This is an example of URI (Uniform Resource Identifier), and is already an active research topic for V2X networks as reported for example in [45] [46] [47] [48]. However in V2X networks the technology trend is to solve this problem using the Information Centric Network (ICN) approach, which does not seem to fit with electronic control network. The reasons why intra-vehicle distributed control networks would not benefit from the ICN approach are, with reference to [49]:

1. ICN network are intended for the decoupling of sender and receiver, in order to better support the mobility of both agents, this mobility of out of scope in a ECUs-based network;

2. although commercial vehicles network can be dynamic and change their structure, links are supposed to be reliable enough, and a critical fault in the network could be considered as disastrous as to cause the machine to get in a fail silent state.

An alternative, function-oriented approach, seems more applicable. A namespace-based structure of the URI is then desirable, to facilitate maintenance, classification of the resources. However, the resource must be described, in order to be useful for the services and applications.

[4]http://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html

The description concerns the identification of parameters that an arbitrary application can access (i.e. read/write), the definition of functionality that the services can consume. These features can be condensed in a resource descriptor that can be easily implemented in a XML file, perhaps as an extension of existing standards like Agro-XML, also called ISO-XML. Hence, at this point any resource is defined by its URI and its descriptor; one ECU, or web-service (or, adhering to this paradigm, one resource) can exploit another resource after consuming its descriptor (in a WSDL fashion), but often can be hardcoded in the functionality of the requesting resource, as if it had a software component very similar to a driver that embeds the means of accessing to a resource meaningfully and correctly. The concept of exploiting resources without acquiring the descriptor from the resource itself leads to a potential enhancement to the RaaS paradigm, which is decomposition of the descriptor in basic sub-resources that do not need further description. This is the same ambivalence found in the REpresentational State Transfer (REST, [50]) architecture and the Simple Object Access Protocol [51], which are commonly used for web services. The single application does not need to be aware of all other applications which are running on the machine (or at least all which can interact with the application) but the relationship is managed by a supervisor. It is important to note that, according to reference standards like SAE J1939 and ISOBUS, the network traffic is currently artificially limited to very low rates (e.g. for Transport Protocol, Broadcast Announcement Message and Extended Transport Protocol), which leads to a very poor resource utilisation, but such a very conservative policy is needed to have a minimum control on the timing of the communications on the CAN bus. This opens a new problem, which is that of resource allocation, and can be solved as an optimisation problem. RaaS can be thought as the common ground for any application of the In-Tractor Cloud paradigm. It gives a planar abstraction of the network and any agent can interact with it, just by accessing it. The namespace depicted gives means to access the resources in a simple, yet efficient and eloquent way. This namespace is used to create a scheme of exploitation: one resource, called Requester Resource (RR), access/exploits another resource, called Exploited Resource (ER). It is important to note that for safety and security purposes this resource plane must not be accessible by any agent, in both its wholeness and its details without due authentication and authorisation.

Authentication/Authorisation-as-a-Service (AuthaaS)

Engineering safety- and security- critical (s&s) services is a topic of topmost interest in designing a complex system as a heavy-duty vehicle. AuthaaS follows the paradigm of security-by-design [52] and the lemma that there is no safety without security [53]. Anyway Authentication and Authorisation (A&A) are necessary but not sufficient to ensure a safe and secure network and other policies and services must be adopted. The IT world has plenty of services and

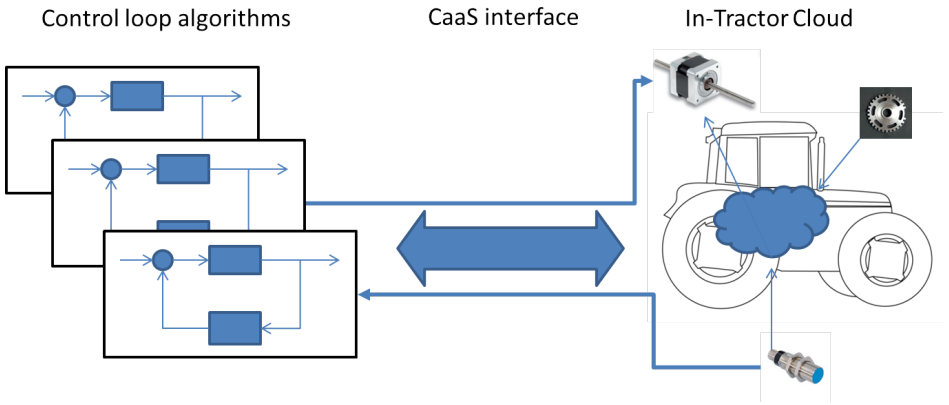


Figure 3.7: Architecture of Control-as-a-Service.

technologies that provide A&A protocols at different ISO/OSI levels that can be adopted in order to create the In-Tractor Cloud, for example:

1. IEEE 802.1x for a port-based Network Access Control,

2. IEEE 802.1q (Virtual LAN, VLAN) for network separation,

3. IETF RFC 4511 (LDAP) for account management and resource access control.

This service, that is currently extremely limited and difficult on SAE J 1939 and ISOBUS networks, as described in section 3.3.3, appears right now as a fundamental and necessary service. In fact on ISOBUS study and standardisation working groups, there is an ongoing effort to introduce cryptographic authentication in the AEF working group for ISOBUS Automation, also called Tractor Implement Management (TIM).

Control-as-a-Service (CaaS)

Up to this point the In-Tractor Cloud has been described in order to delineate the essential parts of it. However the essential duty of a distributed control network is to uphold the constitution of the necessary control systems for the applications to work. CaaS works on two interfaces, one towards the real control system, one towards the other Services, as shown in Figure 3.7. The former re-creates an insulated environment, in which control algorithms, actuators and sensors are resources reflecting the RaaS paradigm. The latter exposes the control system as a resource itself that describes the resources it is exploiting. Exploiting CaaS means that a certain amount of resource must be reserved, eventually with real-time, throughput and reliability requirements for each active control loop.

Monitoring-as-a-Service (MaaS)

MaaS in action will be described in details in the Rear Camera use case in section 3.4.3, and is a direct application of the RaaS paradigm. MaaS can be described as a composition of services, or "service of services", with which the whole In-Tractor Cloud is administered and monitored. MaaS has full authority to reserve resources (e.g. throughput in a specific portion of the topology or a given amount of computational power) to applications, in relation to their priority and criticality, as well as remove previously assigned resources from low-priority services. The priority of a service over its resources is a concept already described in [43]. Many factors concur to create a figure of priority of the service. However, when a given, high priority service is requested by the user, the cloud must respond in a sound and consistent manner: lower priority services must yield part or all of the resources currently held. Even if the low priority services may not be responding or willing to yield resources to other services. MaaS must be capable to relinquish the resources by forcing them to be allocated to the high priority services, preempting lower priority services. The problem of service preemption is analogous to that of a preemptive task scheduler with priority, but executed at network level. In fact it can be seen as a preemptive network scheduler.

Safety as a Service (SafaaS)

A desirable property of a service, be it a resource or an application, is its Performance Level in terms of safety, or safety as a Service (SafaaS). In this context, the safety is intended as the ability to avoid physical injuries to persons even in the case of failures, and it has been described in detail in chapter 2. This approach to safety is in line with the black channel principle, and is well established for example in the openSAFETY stack [54]. For example, a communication channel can additionally offer safety as a service, if employing adequate measures. Alternatively, a computation service can offer a safe variant if implemented by a safe hardware, according to the requirements presented in section 2.1.3. This will of course dictate how temporal and spatial allocation of the resources is done; for example the same service can require the safe variant of the used resources, if required for a safety operation. An example can be that of a rear monitoring camera, and is detailed in section 3.4.3.

Reliability as a Service (ReaaS)

Similarly to SafaaS an additional property of the services can be its degree of reliability (ReaaS). This is closely related to the application, which can require a high reliable control loop, if willing to limit other performance indexes, for example the time period of the control loop. Another realisation of ReaaS can be a computation service able to migrate from one

physical ECU to another, implementing a fail-over operational mode, to have a guaranteed availability of the service. The last example can be represented by a communication service able to transmit information at different level of reliability, adding controls and increasing the needed bandwidth with respect to a lower reliability degree offered for different information classes.

3.4.3 Example of use-cases

In this section two applications of the In-Tractor Cloud are presented, regarding precision farming, a rear monitoring camera and the application of Safety-as-a-Service.

Precision Farming

Precision farming is one of the fields that are pushing for high-speed networks. The control system of a seeder/sprayer or a generic implement ECU can require an amount of resources that increase exponentially as the number of components increase. Furthermore, the complexity of such control system can be challenging. Finally, to be able to operate at high speed, a certain amount of calculations has to be done periodically, even at intervals of 10 or 100 ms, which is a quite short period for the typical hardware used, so the computational power of all the ECUs on the seeder should be increased. A XaaS approach would instead use a remote computation resource, provided as a service. This brings essentially two advantages:

- The ECUs on the seeder can be smaller, hence cheaper;
- There can be various operating modes, depending on the computational resources available, but all of them can be realised with the same seeder.

The remote computation service can be provided by the tractor itself, so the overall cost of the implement can be reduced. Furthermore, the same implement, or the same ECUs, can scale from small machines to big machines, handling the control complexity in an efficient way. Finally, extending a service-oriented implement can be naturally done by plugging a new service, which can use computational and network resources already available.

Rear Camera

In agricultural environment it is becoming always more frequent the use of cameras to monitor areas around the vehicle that the operator is not able to see directly; this functionality is normally called indirect and assisted vision. The rear camera use case is based on RaaS where the network resources are shared between devices and functions with different priority, criticality and in general requirements. There will be a kind of supervisor of this resource pool

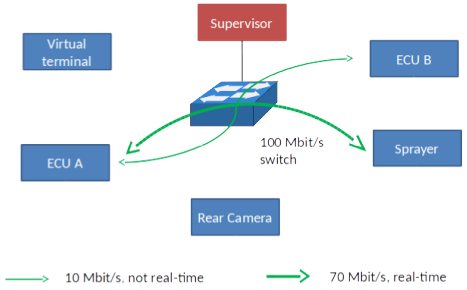
that regulates the use of network resources, which in an Ethernet network can be identified as a switch. The supervisor can force allocation or removal of resources to or from a certain device; for this to happen, the network infrastructure must support some sort of partitioning, e.g. IEEE 802.1 AVB/TSN. In the general case, there are several devices that share the same network infrastructure. The network usage policy requires that each device that needs to use the network will have to ask the supervisor to grant it, for example, a certain bandwidth for its communication, together with a priority; authentication may be needed for certain priorities or resources. At this point the supervisor has several opportunities depending on the current network situation:

1. **Bandwidth available** it is the easiest case in which the supervisor grants to the device the required bandwidth;

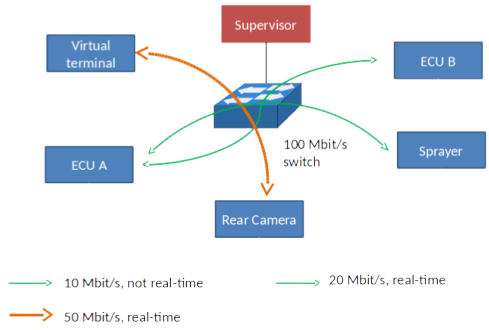
2. **Bandwidth not immediately available but enough bandwidth can be freed** there is not enough bandwidth to satisfy the request but there is traffic with a lowest priority that can be limited in bandwidth. The supervisor will force low-priority devices to relinquish the bandwidth. The devices will have the possibility to stop the communication or to reduce their bandwidth.

3. **Bandwidth not available and no bandwidth can be freed** the operation cannot be executed. This may happend because higher priority application already use all of the available bandwidth. If it is a critical operation then a dangerous condition could result; for this reason, a resource planning must be performed beforehand to ensure that the supervisor is always able to allocate the resources for critical applications.

This is only an example of policy that can be implemented on a supervisor, but more sophisticate examples are possible.
As a concrete example, suppose that the initial situation is the one reported in 3.8(a) where ECU A communicates with the sprayer using 70 Mbit/s with of bandwidth and with ECU B using 10 Mbit/s. Those traffic flows have a bandwidth that has been allocated by the supervisor, which is aware of the current network usage. At a given time the virtual terminal and the rear camera need to establish a communication, for example because the reverse gear has been engaged. They require 50 Mbit/s to transfer the video stream and a high priority considering that the camera is performing a safety critical function, that is to transmit images of areas where the operator has no visibility. The rear camera will send the bandwidth request to the network supervisor. Considering that there are already two traffic flows on the network, there's not enough bandwidth for the video flow, so the supervisor begins to free the low-priority resources and reallocate the required bandwidth. Several scenarios can occur:



(a) Without rear camera.



(b) With rear camera.

Figure 3.8: Vehicular network traffic in the rear camera use case.

1. ECU A-to-ECU B or ECU A-to-Sprayer communication can be interrupted. The supervisor interrupts one of the traffic flows, enough bandwidth is available to be allocated for the video stream and the end-user is informed of the loss of service.

2. No communication can be completely interrupted but both of them can have a bandwidth reduction in order to free enough bandwidth for the rear camera communication. In this case, the bandwidth reduction will be negotiated between the sprayer and the supervisor, in order to allow the higher-priority rear camera to operate correctly. This is the case illustrated in Figure 3.8(b).

Major voting

In agricultural domain most of the safety-related systems are fail-safe and not fail-operational. Here the application of XaaS can favour cost-effective and future-proof products, although the resulting system integrity or availability may not increase but even decrease.
As an example, consider an hardware Category 2 described in section 2.2.1. The main idea is to use the RaaS paradigm to build a Remote or Virtual SMC. Considering as an example

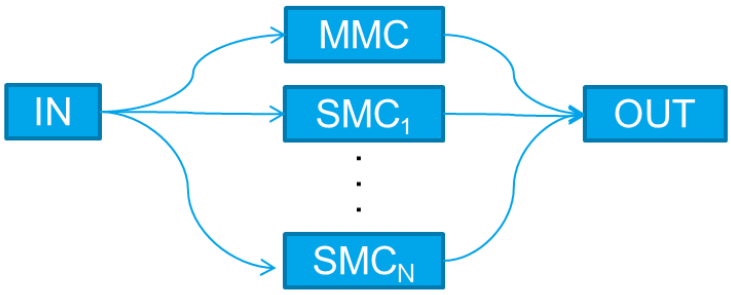


Figure 3.9: Architecture of Safety-as-a-Service related to major voting.

a service like Amazon's EC2 it will be possible to create several virtual elaboration units in order to have a dedicated virtual SMC for each ECU that requires it. In such a way it will be possible to reduce costs providing a virtual SMC if a remote efficient and reliable method to de-energise the output is available (provided this is the safe state), controlled by the virtual SMC. In any case, the method of reaction and the place where the decision will take place depends on the system architecture; it can be for example in the same ECU that is controlling the actuators, but also in a remote ECU, for example controlling a load dump valve for hydraulic power enable, or other solutions for both electrical and mechanical power enable and control. Furthermore, not every ECU in a machine needs to be able to act as SMC, only one (or more) specialised ECUs are sufficient.

If we consider for example to have four control units in the vehicle, the classical implementation would have four dedicated SMC, the cloud implementation can physically have just one microcontroller, used in a most efficient way to provide a Virtual SMC to each ECU.

It is possible to think to an evolution of this architecture where the virtual SMC, rather than performing a fixed elaboration on a pseudo random number, receives also the algorithm to execute through the communication bus (for example using a independent platform executable function). In this way it will be able to perform even more efficiently the diagnostic tests and, in case of faults in a single control unit it can reallocate the computational resources in order to make the functionality that would be interrupted go on. Extending this idea to multiple SMC, this becomes a major voting mechanism, pictured in Figure 3.9. Major voting, in fact, is almost never applied to real control units in agricultural applications, because it would require at least 3 microcontrollers in each unit. This is often referred to as a 2oo3 system (2 out of 3), but more advanced voting systems can be implemented in a flexible way, e.g. 2oo4 (2 out of 4), which ensures availability in case of a single fault [55].

Exploiting this architecture will make easier and low-priced to implement major voting mechanisms because it will only be necessary to allocate an additional virtual SMC for the

control units that require this mechanism. For example, it could be allocated only during the execution of particularly risky applications or tasks and freed when the application ends. Finally, the flexibility of implementation will be greatly increased. Nowadays a control unit designed to be for example an hardware category 2 can not be modified without a complete re-design of the control unit. Using RaaS it will be possible to simply add one or two virtual SMC to the control chain, without hardware modifications, under specific constraints related to actuation characteristics.

3.4.4 Future Perspective

This section has introduced the In-Tractor Cloud paradigm, a revolutionary architecture for tractor and vehicular networks, which allows both new applications and use cases and to overcome the current limits of low-speed CAN-based networks. This architecture is enabled by high-speed networks such as Ethernet and its automotive-grade physical layer IEEE 802.3bw. In this context, safety can be seen as a very special kind of service, and integrated into the In-Tractor Cloud architecture. According to this vision, additional needs for both applications and users will be (see [42] from page 81):

- increased consistency in how functionality and data is represented
- reduced dependencies between units of solution logic
- increased opportunities to combine units of solution logic into different configurations
- increased availability and scalability.

There is a number of further services which can be useful for in-vehicle networks, for example Time Synchronization as a Service (TsaaS), Task Synchronization as a Service (TasaaS), Queueing as a Service (QaaS). This will lead to the implementation of a new generation of Task Controller, Sequence Controller and File Server, that will become the “Task Control as a Service”, “Sequence Control as a Service” and “File Storage as a Service”. A real implementation of these services could be seen as the first steps towards a standardized Service Contract, Service Discoverability and Service Autonomy for an implementation of next generation ISOBUS.

Chapter 4

Wireless Sensor Networks - State of the Art and Applications

This chapter presents in more detail some of the technologies currently used to build WSNs. The list is by no means exhaustive, since the number of competing technology is way too high to be analysed here. However, some of the most well known are considered here, and a brief overview of their strength and weaknesses is given, with reference to some example application. Finally, an overview of higher protocol layers is given, as well as some hardware platforms developed by either the academia or the industry.

4.1 Overview

Wireless Sensor Network are quite different from classic consumer networks, based on IP and Ethernet. Although IP can now be used on top many WSN technologies, the underlying semantic at link layer is quite different from a typical Ethernet or Wi-Fi network. In some networks the communication is not even bidirectional, because all that is needed is to read some raw value. In this case re-configuring or updating the device of course require a physical intervention, but there are applications where this is economically convenient, such as TPMS applications, where a wireless sensor is drawn into the tire plastic, with a battery capable of lasting several years.

One of the most important requirements is the low-power, in contrast to the possibility of wired network or even IEEE 802.11-based to be almost always powered-on. This has a huge impact on the communication semantic at layer 2 in the ISO/OSI stack since the transmission or receive operations are the most energy-consuming operation on a typical WSN sensor; it is then convenient to define a master and a slave role. The master is the device with higher energy supply, typically a base station, often connected to a permanent energy source, while the slave device is a battery-powered sensor, and often the battery is limited in capacity

because of application constraints like physical dimensions, cost or weight. For example, in the IEEE 802.15.4 standard, the MAC layer defines two different communication semantics for *uplink* (slave-to-master, or sensor-to-station) and *downlink* (master-to-slave, or station-to-sensor). Uplink communication can happen at any time (for non-beaconed networks) or in some pre-defined time periods (for beaconed networks) but in both cases the base station is always active waiting for the packets from the sensor, without necessarily knowing if there actually are some packets to be received. Sensors of course transmit the packets to the station only when they have some meaningful data to send. The sensor however periodically turn on the transceiver if participating in a beaconed network, to receive the beacon packet. On the other hand, downlink communication must be agreed between the station and the sensor, so the sensor can turn on the transceiver at the right time and only if there is something to receive. Considering for example IEEE 802.15.4, this can be accomplished either by a flag in the beacon packet, signalling a pending downlink packet for a specific node, or by the sensor periodically polling the station to check for pending packets. This asymmetry is necessary to save as much power as possible on the sensor node, by turning on the radio transceiver only when needed.

While WSN have been created for communication networks with a low duty-cycle, that is where the time spent by a sensor in low-power mode is much longer than the time spent transmitting or receiving packets, WSN can present some advantages also on networks with a medium or high duty-cycle, when compared to other wireless technologies like Wi-Fi or Bluetooth. Compared to Wi-Fi, for example, many technologies allow for complete control over the link layer, and this is particularly useful for real-time networks, and also for research and development of new protocols. This is also possible because of the low-bandwidth of the wireless link, which in turn allows for communication timings which can easily be handled by a microcontroller with reduced computational power. As a consequence, protocol extensions have been proposed, like IEEE 802.15.4e [56], specifically designed for industrial applications with real-time requirements. The possibility to modify the MAC layer also made it possible to implement, at least partially, the modifications described in chapter 5. Last but not least, WSN transceivers have typically a lower cost than other wireless technologies like Wi-Fi.

4.2 Current Technologies

In this section different network technologies are considered. A network technology is here considered as composed of a physical layer specification, as well as some protocols for upper layers (comparable to link and network layer in the ISO/OSI model). This reflects the fact that in many wireless sensor networks a specific protocol stack is tied to the underlying

physical layer, either because defined in a standard such as IEEE 802.15.4 or because part of a proprietary stack, such as LoRa.

4.2.1 IEEE 802.15.4

The IEEE 802.15.4 standard [57] MAC level is designed for low-power networks, mesh, tree and star topology, and some low-latency modes have been developed [56], allowing for a worst-case latency of the order of tenth of milliseconds. It is generally designed for bidirectional short/medium range communications, up to 100m in some cases. Many physical layers are defined in the standard, including both narrow and ultra-wide band modulations. While the MAC layer provides a modified AES-CCM algorithm for encrypting data of arbitrary length, it does not provide a method for key exchange and management. This is currently under development by the Task Group 9, and will probably result in the IEEE 802.15.9 standard. The authentication of the device is application-dependent.

IEEE 802.15.4 – sub-GHz

The IEEE 802.15.4 standard [57] defines operating modes for the 780, 868-915 and 950 MHz bands, with narrow- band ASK, BPSK, O-QPSK and GFSK modulations and a bit rate from 20 to 250 kbit/s. The use of sub-GHz carrier frequency allows to reach in some cases better performances than with higher carrier frequency, especially in presence of obstacles or barriers, as reported in [58].

IEEE 802.15.4 – 2.4 GHz

In [57] a 2.4 GHz phy is defined, with O-QPSK and CSS modulation and a bit-rate from 250 kbit/s to 1000 kbit/s. The performance may in some cases be worse than UHF modulations, both for the higher frequency and noise. This is the PHY layer used in ZigBee networks, where some key management support is also present.

IEEE 802.15.4 – UWB

The Ultra Wide Band technology is a relatively recent modulation technique, with 500 MHz and 1 GHz modulation bandwidth; it allows a precise estimation of the distance between two WSN nodes thanks to the ability to measure the time of flight of the data packet, through a 2-way or 3-way handshake. This measure has also been standardised in IEEE 802.15.4. Furthermore, it is resistant to multipath fading, making it suitable for cluttered indoor environments.

4.2.2 Bluetooth LE

The technology commonly known as Bluetooth operates on the 2.4 GHz ISM band, and is maintained by the Bluetooth Special Interest Group. While originally designed for Body Area Networks, with high data rate, the 4.0 revision introduced the Low Energy mode, specifically designed for Wireless Sensor Networks, with low power and low data rate. Based on an attribute system, a Bluetooth LE device can expose sensor values as a service. While it provides methods for encrypting the data, the key management has still to be done in part by the application, as well as device authentication, although there is built-in support for bonding and pairing of devices.

4.2.3 Micro.SP

Micro.SP is a proprietary technology developed by STE S.r.l and is designed to be extremely low power, at the cost of being unidirectional. The protocol used for sensor reading is very simple, and there are no MAC layer or access control mechanisms. The PHY layer is based on a PPM (Pulse Position Modulation) modulation in the 433 MHz band, designed for low data rate and short range communications. Currently it does not support any form of encryption or authentication, only an ID field is present.

4.2.4 LoRaWAN

LoRa is a proprietary technology developed by the LoRa Alliance, and is designed for long range communications in a wireless sensor network. It provides a dedicated physical layer with low data rate and a complete application layer, with a master-slave architecture. It can be used on different ISM bands, such as 433 and 868 MHz, and provides support for both authentication and encryption of data.

4.2.5 ANT

ANT is a proprietary technology managed by ANT Wireless, designed for low-power, low data rate and short range communication and sensor reading,. It provides a dedicated physical layer, based on a GFSK modulation, with low data rate and a complete application layer, allowing for a star, mesh and tree network topology. It is designed to be used in the 2.4 GHz ISM band, and provides support for both authentication and encryption of data.

4.3 Example of applications

Smart sensors are today used in a wide range of applications and scenarios, from industrial and home automation to very short range applications such as medical and health monitoring. This

section will consider wearable applications as an example; here the sensor node are designed to be worn easily in order to perform one or more monitoring tasks, such as patient health monitoring, elderly people tracking and fitness measurements. For this reason, several body parameters such as temperature, blood pressure, heart rate, muscle activity, etc. are intended to be measured and analysed locally [59]. This is the main difference between the traditional wired sensory systems where the processing were done in a central unit rather than the current distributed processing. This is due to the recent advancements [60] in microelectronic systems which led to develop tiny sensor nodes capable not only of sensing, but also processing and collaborating with other sensors to fulfil the overall system's goal. In this context, human activity monitoring has recently drawn the attention of the researchers to propose various systems and application scenarios. For this reason, a comprehensive study of application requirements is needed in order to come up with a feasible and robust system which performs well in different environments. According to [61], there are 8 parts that are required to be considered for a typical wearable system:

1. types of sensors to be used;
2. type of wireless protocols to be employed;
3. monitoring of activities to be considered;
4. methodology to determine activities or extraction of important features;
5. design and development of small, light-weight, powerful and low-cost smart sensor nodes;
6. harvesting of energy for normal operation and communication;
7. ability to be used with the present day mobile devices;
8. flexible to configure the system for a new user without much difficulty.

Currently, there is a large number of research going on considering those 8 parts. For health care applications, accessibility, flexibility, security, remote data acquisition, personalised services and automatic decision making are also widely considered in the literature [60]. As a result, the first stage is to identify the types of the sensors. This is highly dependent on the application and the main system's goal. However, the wearable sensors can be grouped in two categories, depending on the wearable technology: electronic skins and clothing-based sensors. Recent technologies has led to propose various smart and wearable sensors based on Micro-electro-mechanical Systems (MEMS). Examples of the recent technologies in wearable sensors can be found in [62] and [63]. Due to the wireless nature of smart wearable sensors, various parameters are required to be considered while designing the system infrastructure,

such as latency, energy efficiency and reliability. This is very crucial especially for the medical and high risk application where the network robustness is a key factor. A complete analysis of the mentioned parameters is given in [64] where the robustness and latency are discussed for short range body sensor networks. Another characteristic of the wearable sensors is the dynamic channel property, that is, when the body moves, the channel characteristics are also changing and the system can use different channels at a specific time. This will result in a fading effect in the channel which has a direct impact on the reliability of the communication. The IEEE 802.15 Task Group 6 proposed the standards for the body sensors in [65] where a complete description of the wireless channel modelling is given. For this reason, there is an increasing interest for modelling dynamic channels especially for the wearable sensors. For example, an extensive measurement has been carried out in [66] using a multi-port channel sounder in order to model time-variant and multi-link channel. Several parameters such as fading, link auto and cross correlation are used to model the body movements. With these considerations in mind, a review of some existing applications and scenarios of the smart and wearable sensors is done here. The main focus is on the existing wireless communication technologies in order to give a road-map for selecting proper technology that fits best for the intended application.

4.3.1 Scenario and requirements

In this section a description of the different scenarios considered is provided. After the scenarios description a requirement analysis for different applications will be performed. This sample of applications is going to be useful in order to highlight the considerable differences in the parameters to be considered relevant in the technology selection depending on the environment in which the network will be involved.

The first WSN application is the use in sports, from team to individual ones, like soccer, basketball, hockey, martial arts, gymnastics, swimming and diving. Current technologies are mainly based on measuring and off-line processing; only the visualisation of data is performed in real-time, for example through a Graphical User Interface. A remarkable improvement can be the on-line data processing to allow the provision of real-time feedback, which can be multi-modal, for example both audible and haptic signals, or the inclusion of artificial intelligence techniques, for example in order to have pattern recognition or probabilistic path planning.

The second wireless sensor network application is the security, search and rescue. In this case the parameters that are most useful to monitor are the vital signs of the rescuers. A less obvious set of parameters is related to the scenario where the task is carried out, like descriptions of the areas of the building, the areas of increasing danger, or other elements

that may harm the user. The WSN itself can also improve user localisation systems in areas where the GPS signal is not available. An example may be that of a fire in a building where a rescue team needs to go in and search for survivors. In this case it can be useful to have information not only about the structure of the building but also about walls that can be safely tore down or areas where it is more likely to find people. Complementing this information with an approximate location of the rescuer is very helpful. Also the application of machine learning and pattern recognition may allow to anticipate or prevent an hazard to the rescuer. The third application is health care and, in particular, home care. In this case the WSN can be used to grant a monitoring tool for patients that requires a very strict observation and care after an hospital discharge. A similar solution will permit a 24/7 home assistance creating a safe environment which respect person's lifestyle without imposing any change. A different home care scenario can arise from silver care, where monitoring elderly people can help preventing harms from falls, or monitor existing diseases, for which a precise tracking of the person's movements can prove very valuable.
The parameters used in the technology evaluation arethe following:

- Latency
- Power Consumption
- Range
- Bidirectional communication
- Cost
- Support for encryption, authentication and key management
- Support for localisation

All these parameters will have a different importance in the evaluation of the best WSN choice depending on the application considered. Other criteria could be added to the comparison, for example antenna gain and size, throughput, fading resistance and many more; however too many parameters could lead to a difficult weighting of the characteristics and the subsequent evaluation of the technology. This choice of parameters is believed to be simple and general enough for the three scenarios considered. Latency is important only for those applications that needs to perform a real time analysis of the data. In the scenario described the only one that needs this characteristic is the sport application. The rescue application and the medical one works well also if the latency is high (it is considered an high latency a latency of about one second). The power consumption is important for those applications where the WSN has to work for a long period of time without the a battery recharge. An example of

such an application is the medical home care in which the patient should be continuously monitored for days or weeks. This characteristic is not very important in applications with a short duration such as rescue or sport, where the use of the WSN is limited to some hours or at worst a whole day. The range is important for those applications that has to monitor the data from a long distance. The only application in which this parameter is not relevant is the home hospitalisation in which a data collector inside the house can be provided. In this case it is sufficient to cover the area where the patient lives, which can be restricted to the house. A bidirectional communication is useful mainly if it is necessary to dynamically configure the sensor because of the variation of the environmental parameters. The sport application can request this kind of operation due to the different kind of sports in which the WSN can be applied, or different training programs and exercises for the same sport. It could also be useful to have a secure handshake, if there is a need for a key setup procedure, otherwise a private key can be programmed off-line on the sensor in a non-volatile memory. The cost is a parameter that is relevant, to a different degree, in every application scenario. In sport the devices must be accessible to consumer users, in medical and rescue applications there is a limited budget available to buy new devices, so to grant the service to as more people as possible the cost has to be considered. However a higher cost may be acceptable, for example because there is the possibility to buy large quantities of wireless nodes at once. The support provided for authentication and encryption is essential where there are security or privacy concerns. Given that a wireless body network will most likely transport personal data, the need for encryption can be assessed in every application, as well as the need for authentication. The means by which these needs are satisfied however can vary greatly depending on the technology used, and will in some cases limit the possibility of applying cryptographic protocols. Also additional requirements, such as security threats, or strict legal requirements, such as health legislation, could demand for higher level of encryption or some form of authentication, or pose some constraints on key management. These requirements will be briefly referred to as crypto requirements. Finally, the support for spatial localisation of the user wearing the wireless body area network, although desirable in every application scenario, can have different degrees of precision required, for example in the range of meters vs centimetres. While a high precision is desirable in all three scenarios, in the health care it can have the greatest impact, in terms of new applications that become possible. A summary of the parameters and their importance is given in Table 4.1.

4.3.2 Technology Evaluation

In this section each technology is evaluated with regard to the criteria established in the previous section. For each criteria, the performance achievable only relying on the protocols and

Scenario	Metrics						
	Latency	Power	Range	Bidir	Cost	Crypto	Localisation
Sport	*	***	****	*****	*	***	***
Security Search Rescue	*****	***	****	*	**	****	***
Health-Care	*****	*	**	**	**	*****	*****

Table 4.1: Example of requirements of a wearable wireless sensor network for different application scenarios. *****High relevance, **Medium relevance, *Low relevance

characteristics of the technology itself are considered, rather than the ability of a technology to fulfil a specific requirement by means of additional protocols at the application layer. This is to avoid confusion in evaluating for example the latency, where a lower value is given by higher performance, or crypto, where it can be relatively easy to add the missing protocols at the application layer. Also power consumption is difficult to compare between different technologies, since in many cases it is heavily depending on the specific configuration of the network and the data rate. Here the evaluation of power consumption is based on the values reported on the datasheet of a commercial SoC or transceiver. A summary of the evaluation for all chosen technology is reported in Table 4.2.

IEEE 802.15.4 - 950 MHz

As an example, the Freescale KW0x SoC [67] is chosen. It consist of a MCU with an embedded narrow-band radio transceiver, which carrier can be set to 290-340 MHz, 424-510 MHz, and 862-1020 MHz frequency bands. It supports, among other combinations, the 950 MHz GFSK operating mode of IEEE 802.15.4, with a bit rate of 100 kbit/s. It also support non-standard modulations, up to a bit rate of 600 kbit/s. According to the data sheet the current consumption is 20 mA when transmitting at 0 dBm. Since this consumption is stable across the VDD range, it has a minimum instantaneous power consumption of $1.8V \cdot 0.02A = 36mW$ without considering the MCU consumption. This technology has overall low power consumption Supporting the IEEE 802.15.4 standard allows for bidirectional communication and its cost is low compared to other technologies. From experimental measures, the range is approximately 60 m indoor and 100m outdoor.

IEEE 802.15.4 - 2.4 GHz

As an example, the Linear LTC5800-WHM SoC [68] is chosen. It consist of a MCU with an embedded narrow-band 2.4 GHz radio transceiver, compatible with IEEE 802.15.4, with a bit rate of 250 kbit/s. According to the data sheet the current consumption is 5.4 mA during transmission at 0 dBm. The power consumption during transmission is then $3.76V \cdot 0.0054A = 20.3mW$. This technology has overall low power consumption. Supporting the IEEE 802.15.4 standard it allows for bidirectional communication but its cost is relatively high compared to other WSN technologies. From the data-sheet the range is 100 m indoor and 200 m outdoor, with a +2dBi antenna. However different SoC from different vendors can have quite different performance; for example in [69] the authors report a much shorter range of 40 m indoor and 70 m outdoor.

IEEE 802.15.4 - UWB

As an example, the DecaWave DWM1000 [70] module is chosen. It consist of a transceiver with built-in antenna, fully compatible with IEEE 802.15.4 and with some proprietary extended modes, allowing for a bit rate of 110 kbit/s, 850 kbit/s and 6,8 Mbit/s. According to the data sheet the current consumption is 31 mA during transmission at -14 dBm. The power consumption during transmission is then $3.3V \cdot 0.031A = 102mW$. The power consumption is then relatively high, compared with other physical layers defined in [57]. On the other hand, the achievable range is relatively higher, considering the lower transmission power.

Bluetooth LE

As an example, the Texas Instruments CC2540 SoC [71] is chosen. It consist of an MCU with an embedded Bluetooth 2.4 GHz radio transceiver, which is based on IEEE 802.15.1, with a bit rate of 1 Mbit/s. According to the data sheet the current consumption is 5.4 mA during transmission at 0 dBm. The power consumption during transmission is then $3V \cdot 0.027A = 81mW$. This technology has overall medium/low power consumption. Bluetooth allows for bidirectional communication but its cost is relatively high compared to other technologies, especially considering the need to pay a fee to advertise the use of Bluetooth itself.

Micro.SP

The Micro.SP technology developed by STE [72] is chosen as an example of a proprietary wireless sensor network optimised for short range and extremely low power consumption. It consist of an MCU with an Micro.SP transceiver operating in the 433 MHz ISM band,

with a bit rate of 10 kbit/s. According to the data sheet the current consumption is 35 mA during transmission at +14 dBm. The power consumption during transmission is then $3V \cdot 0.035A = 105mW$, with an average current of less than 400 uA. This technology has overall very low power consumption, since the average power consumption is $3 \cdot 400 \cdot 10^{-6} = 1,2mW$. However this technology allows for unidirectional communication only, in order to keep the power consumption at this level. Its cost is low compared with other technologies.

LoRaWAN

The Microchip RN2483 module is an example of a LoRa-enabled transceiver [73], and is chosen as an example of a proprietary wireless sensor network optimised for long range communications. It consist of a MCU with a LoRa-enabled transceiver operating in the 433 or 868 MHz ISM band, with a bit rate of 5468 bit/s. According to the data sheet the current consumption is 22.3 mA during transmission at +0.4 dBm in the 868 MHz band. The power consumption during transmission is then $3V \cdot 0.0223A = 66,9mW$. The technology is optimised for long-range communications, up to 15 km in suburban area and 5 km in urban area, according to the data-sheet.

ANT

The Nordic Semiconductor nRF51422 SoC [74] is an example of an ANT-enabler system, and is chosen as a sample implementation. It is a mixed Bluetooth-ANT module, and can be used with a data rate up to 2 Mbit/s. According to the data sheet the current consumption is 10.5 mA during transmission at 0 dBm; the power consumption during transmission is then $3V \cdot 0.0105A = 31.5mW$.

4.3.3 Technology selection

In this section the use of the aforementioned technologies is discussed for each application scenario, evaluating their suitability and the effort in adapting to the specific needs. A summary of technology fitness is reported in Table 4.3. For the sport scenario, the best fit is found with the IEEE 802.15.4 – 950 MHz case. The main reason for this is the capability of having low latency for sensor reading, which is a key enabler for future application of real-time performance monitoring in the sport field. In some cases a Bluetooth technology could be sufficient, for short range monitoring, but this will restrict the applicable sport use cases. The support for encryption only in current standards is considered to be more than enough for the security needs of performance monitoring, since a sensor could be assigned a specific static encryption key. The security/search/rescue allows different technology to be considered. The most stringent requirement is the range, so the choice could be either IEEE 802.15.4-950 MHz

Technology	Metrics						
	Latency	Power	Range	Bidir	Cost	Crypto	Localisation
IEEE 802.15.4 950 MHz	*	**	***	*****	*	***	***
IEEE 802.15.4 2.4 GHz	*	**	**	*****	**	***	***
IEEE 802.15.4 UWB	*	****	***	*****	*****	***	*****
Bluetooth LE	***	***	*	*****	***	***	*
Micro.SP	*****	*	*	*	**	*	*
LoRaWAN	*****	***	*****	*****	*****	*****	*
ANT	***	**	**	*****	**	*****	*

Table 4.2: Summary of performance level characteristic of WSN technologies.

or LoRa. While the former is attractive for its low cost, the latter has the main benefit of allowing a longer range communication and including protocols for authentication and data encryption. A mixed-network approach could be possible, if many sensor are requested, using a Micro.SP network and a collector for long-range communications. In the health care scenario the most stringent requirements are low power consumption and high security level. In the former case the Micro.SP technology offers the best performance, but at the same time it is also the technology with less built-in security mechanisms, since data are normally sent not encrypted. However it may very well be convenient to add the missing security protocols at application layer, provided that no dynamic reconfiguration of the sensors is necessary. This is often the case, since bio-sensors have a well-defined role and calibration. However also a mixed-network approach is possible, to compensate for the Micro.SP short range. In this case, IEEE 802.15.4 UWB would allow for very precise localisation, which can be necessary for some silver care application.

4.4 Higher Layers

It is only recently that WSN implement to some degree the end-to-end principle. The typical infrastructure consisted of an application level gateway to interface external networks with

Scenario	Technology						
	IEEE 802.15.4 950 MHz	IEEE 802.15.4 2.4 GHz	IEEE 802.15.4 UWB	Bluetooth LE	Micro.SP	LoRaWAN	ANT
Sport	*	***	****	*****	*	***	***
Security Search Rescue	*****	***	****	*	**	****	***
Health-Care	*****	*	**	**	**	*****	*****

Table 4.3: Scenarios and technology suitability

the WSN, which was designed specifically for each single application, because an architecture similar to the Internet's was thought to be unnecessary. This was initially the main design methodology and is still used today on some applications. The result is a great number of different technology, incompatible one each other, which in the majority of cases have similar characteristics and that are the result of sparse efforts.

4.4.1 IPv6 for WSN

The idea of using the Internet Protocol on WSN started to appear following the "Internet of Things" trend. While the use of IPv6 instead of the lighter IPv4 slightly degrades the performance of a WSN, due to an increased raw packet size, its advantages like the huge number of addresses and anycast compensate this performance degradation. The effort of using IPv6 on a WSN resulted in the creation of the 6LowPAN IETF Working Group, which created 3 main specifications, RFC4944 for IPv6 over IEEE 802.15.4 [75], RFC6282 for header compression techniques [76] and RFC6775 for neighbour discovery optimisation [77]. Furthermore, the 6tisch IETF working group developed an optimised architecture and protocol for IEEE 802.15.4e TSCH operating mode which challenges were defined in RFC7554 [78] and are currently a draft specification.

More recently, an industrial effort under the Thread protocol stack [1] aims to include many existing open standards like IETF's, and there is also a reference implementation called OpenThread [2].

[1] https://www.threadgroup.org/

[2] https://github.com/openthread

4.4.2 Routing

A good example of this variety is routing protocols. At some point there was a coordinated effort from IETF to create a common specification for a routing protocol suited to WSN needs, the ROLL Working Group, which created the RPL specification, formalised as RFC 6550 [79]. The routing protocols for WSN are quite different from typical routing protocols like OSPF, OLSR, IS-IS, because they are designed for specific traffic conditions, as illustrated in section 4.1. In a WSN, the majority of data is usually transferred from the sensor nodes to a central station, acting as a collection point, so a routing protocol for WSN is optimised for a Multipoint-to-Point uplink communication. Although direct communication between sensors or traffic from the station to the sensor is usually still possible, it is usually less efficient.

RPL

RPL is a distance-vector and source-routing protocol, designed to operate on top of different link layers, for example IEEE 802.15.4 MAC and PHY layers. It is designed to be highly adaptive to varying network conditions and to offer alternate paths in case the default one is not available. RPL is based on the construction of a Destination-Oriented Directed Acyclic Graph (DODAG), where the root of the DAG is the station node collecting all the data transmitted from the sensors. There can be different instances of RPL in a single WSN, defined by an *InstanceID* and including one or more DODAGs. One node is allowed to be part of only one DODAG in each instance of RPL, but it can be part of different RPL instances. The traffic from the sensors to the station is transferred along the branches of the DODAG, from each node to the collecting point. If a packet has to be sent from one node to another which is not the base station, the packet is first sent to the base station which in turn reroutes the packet along the correct branch of the DODAG tree. However RPL allows also an optimised mode, called storing mode, where the packet only needs to reach the highest common node between the two branches in the DODAG. RPL defines all its control messages as ICMPv6 messages; however the specifications are both vague and usually they are implemented only partially, leading to different incompatible implementations of RPL, for example by WSN operating systems like ContikiOS and RiotOS. For this reason, a reduced version called RPL-Lite has recently been proposed, which tries to force compatibility between different implementations and remove functionality that are rarely used. RPL-Lite is described in [80].

4.4.3 Application protocols

Different application protocols have been standardised at the application level, which handle the reading of the sensors (uplink) and other traffic like node configuration (downlink) or

node-to-node communication (peer-to-peer).

CoAP

The Constrained Application Protocol (CoAP) is the result of the IETF CoRE working group and is formalised as RFC 7252 [81]. It is based on UDP as a transport protocol, and it is designed for Machine-to-Machine applications. It includes methods to discover available resources and observe them, using a RESTful API similar to HTTP. It also supports multicast traffic.

MQTT

The MQ Telemetry Transport (MQTT) is a messaging protocol based on the publish-subscribe paradigm, to be used on top of TCP. Is has been standardised both as an ISO standard (ISO/IEC PRF 20922) and to OASIS. An MQTT-based system requires a central *broker* to receive subscribe events and dispatch publications. There exist a variation called MQTT-SN which can be used on non-TCP/IP networks, like Zigbee.

4.4.4 AMQP

The Advanced Message Queuing Protocol is a message-oriented middleware protocol, covering the aspects of queuing, routing, reliability and security. It is often seen as a super-set of MQTT[3] since it supports almost all the features of MQTT and many more.

4.5 Hardware Platforms

Common hardware platforms for WSN nodes are based on low-power microcontrollers like MSP430, which is by far the most widespread, with an external transceiver. Recently a number of of SoC started to include a transceiver peripheral inside the SoC itself, mainly based on ARM Cortex-M architecture, for example Texas Instruments CC2548. At CNR-IMAMOTER the platform used initially was *Zolertia Z1*, in the context of the FACTOTHUMS project[4]. The Z1 hardware platform is a complete sensor board manufactured by the Spanish company Zolertia[5]. It is composed of a MSP430f2617 SoC which includes various peripherals like ADC, GPIO, SPI, I2C and UART. As a radio transceiver it has a CC2420, connected to the main SoC through SPI. Additional on-board peripherals are an EEPROM memory and a temperature sensor. Various expansion modules exist, containing a lot of different sensors. Zolertia sells

[3] https://lists.oasis-open.org/archives/amqp/201202/msg00086/StormMQ_WhitePaper_-_A_Comparison_of_AMQP_and_MQTT.pdf

[4] https://unit.artov.imm.cnr.it/projects/factothums-factory-technologies-humans-safety

[5] http://zolertia.io/z1

an all-inclusive package including an external antenna and a battery pack. Recently Zolertia developed a new product called *Zoul*, which is based on a more powerful ARM Cortex M3. Commercial WSN nodes are mainly targeted to the consumer market, and as such lack some features desirable in a safe WSN; this is one of the reason to develop a new hardware platform at CNR-IMAMOTER, that will be described in section 6.4.

Chapter 5

Advanced Protocols for WSN

This chapter deals with the communication protocols to be used in a safe WSN, both at the MAC layer and the application layer. For the MAC layer, the most challenging issue is real-time communication. It is a very important issue and for this reason in IEEE 802.15.4e [56] additional operating modes for the IEEE 802.15.4 standard are described. This amendment describes three additional operating modes, LLDN (Low-Latency Deterministic Network), TSCH (Time-Slotted with Channel Hopping) and DSME (Deterministic and Synchronous Multi-channel Extension). TSCH is designed to be agnostic to the network topology, providing only a mechanism for slot synchronisation and demanding higher layers to configure the schedule. DSME is a multi-beacon mode which can have peer-to-peer links, and can be seen as an extension of the GTS mode in [57]. Finally, LLDN supports only a star topology by providing a reduced packet overhead, being designed to achieve the lowest latency possible. In this thesis LLDN has been considered as the best candidate for a low-latency WSN and some improvements are presented, both for worst-case latency and configuration time. Finally, a high diagnostic coverage at the application layer is required. Having a MAC protocol layer with real-time characteristics is essential to control and diagnose timing errors but it is not sufficient, since other communication errors can result in dangerous behaviour and must be detected. With the help of existing standards like ISO 15998 and ISO 25119, a transmission protocol optimised for WSN with high diagnostic coverage is presented.

5.1 Overview of the IEEE 802.15.4e LLDN

The LLDN mode is a beacon-enabled mode where all communications span over a set of time slots, which can be shared between multiple nodes or reserved to a single node. LLDN is designed to work in a star topology network, with a PAN coordinator (PANc) and N nodes, as illustrated in Figure 5.1. In shared time slots, access is provided through a slotted CSMA/CA mechanism described in section 5.1.1.4.4 of [56], enhanced with a RTS/CTS

procedure described in 5.1.1.6.6 of [56], and are signalled by the PANc with a CTS Shared Group packet at the beginning of the slot. In reserved time slots access is provided without CSMA/CA, nor any RTS/CTS procedure. Reserved time slots are then suited for real-time communication since the transmission of a packet, if any, will happen at a defined time in the superframe, while shared time slots are suited for non real-time or in general low-priority communication, since the transmission of a packet is delayed by CSMA/CA and RTS/CTS.

5.1.1 Superframe structure

In LLDN the length of each time slot is multiple of a base time slot (BTS), whose quantity and length is contained in the beacon packet, and is itself a multiple of the PHY-dependent symbol time. Standard LLDN defines four types of slots:

- Beacon time slot (BeTS),
- Management time slots (MTS),
- Uplink time slot (UTS),
- Bidirectional time slot (BiTS).

The BeTS contain the beacon sent from the PANc for synchronisation purposes and to signal the current configuration of the superframe. MTSs are reserved for management data; they may not be present in every superframe, can encompass up to 7 BTSs and are at maximum 2, one for uplink and one for downlink packets. The UTS are reserved time slots, used for data sent from a device to the PAN coordinator. A number of Uplink time slots can be reserved for the retransmission of packets from the previous superframe which have not been acknowledged. A separate Group Acknowledge slot can also be reserved to send acknowledgements, from the PAN coordinator to the devices, in addition to the beacon fields. These two last optional Uplink time slots kinds will not be considered in this work, since the optimisation proposed here do not directly involve them. BiTS are shared time slots, placed at the end of the superframe and are used for bidirectional communication of application data between the PANc and the various devices. Their direction is fixed within each superframe and specified in the beacon.

5.1.2 LLDN Operating States

A LLDN network can be in any moment in one of three states: Discovery, Configuration and Online; the life-cycle of a LLDN network is visible in Figure 5.2. Initially, PANc creates a new network by sending beacons indicating Discovery State; then each device announces its presence and the duration and type of the required time slots to the PANc, using MTSs. In

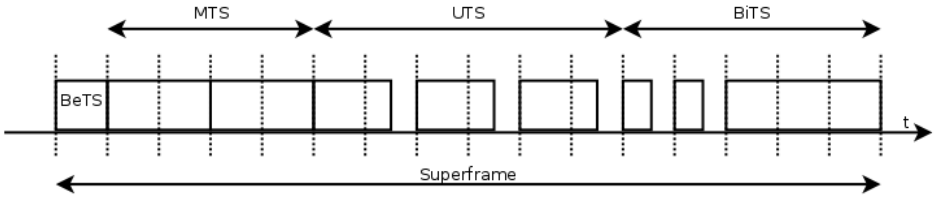


Figure 5.1: IEEE 802.15.4e LLDN Standard superframe format.

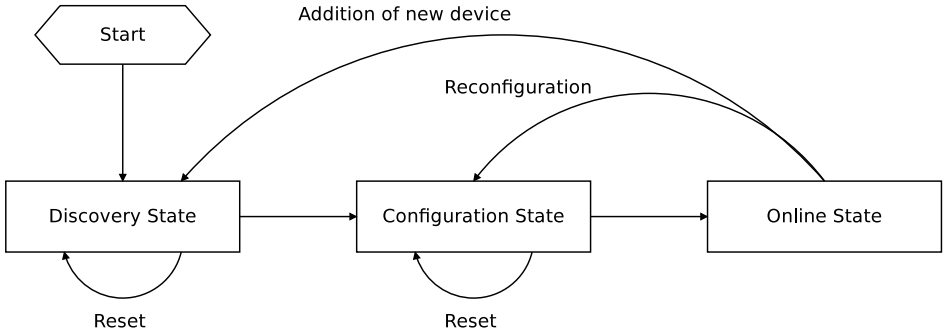


Figure 5.2: IEEE 802.15.4e LLDN operating states and state transitions. Redrawn from figure 34a in [56], copyright ©2012 IEEE.

Configuration State, each node sends its current time slot configuration to the PANc, along with the extended and short MAC address in a Configuration Status packet. The coordinator then replies with a Configuration Request packet, in which it indicates some information, useful for correct management of the superframe. The node accepts the configuration with an ACK frame in the next Management time slot. After all devices have been configured, the PANc signals the Online State, where application data can be sent by the nodes. After every superframe indicating Discovery or Configuration, a PANc can transmit a beacon indicating that every device must reset their discovery or configuration state, restarting the packet handshake. In Discovery and Configuration States, only MTSs are present, while in Online State all of the four types of slot can be used.

5.2 Worst-case latency optimisation

Throughout this section, real-time traffic is also referred to as high-priority, while low-priority traffic does not have real-time requirements. A summary of the notation used is reported in Table 5.1.

5.2.1 Related Work

Though most of the research work regarding bandwidth reservation in IEEE 802.15.4 done so far has focused on the Guaranteed Time Service (GTS) mechanism described in section 5.1.1.1 of [57], evaluating either allocation schemes such as in [82], modified superframe structures

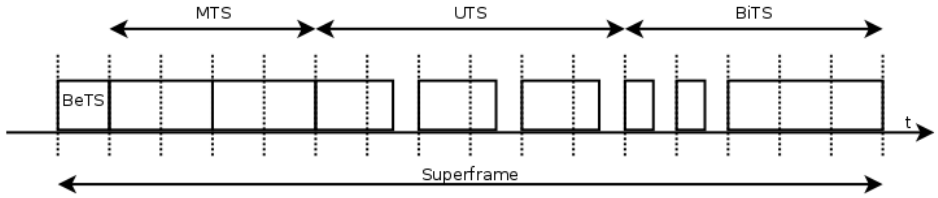
Symbol	Description
N	Number of nodes in a LLDN network, excluding the PANc
n_i^s	Number of reserved time slots assigned to the i-th node
T_s	Duration of a LLDN superframe
T_c	Cycle time, i.e. worst-case latency
T_{BTS}	Duration of the Base time slot
T_{BeTS}	Duration of the Beacon time slot
T_{UTS}^j	Duration of the j-th Uplink time slot
T_{DTS}^j	Duration of the j-th Downlink time slot
T_{BiTS}^j	Duration of the j-th Bi-directional time slot
U_{BeTS}	Number of BTS in a BeTS
U_{UTS}	Number of BTS in a UTS
U_{DTS}	Number of BTS in a DTS
U_{BiTS}	Number of BTS in a BiTS
k_{UTS}	Number of UTS in a superframe
k_{DTS}	Number of DTS in a superframe
k_{BiTS}	Number of BiTS in a superframe

Table 5.1: Summary of the notation used throughout this chapter.

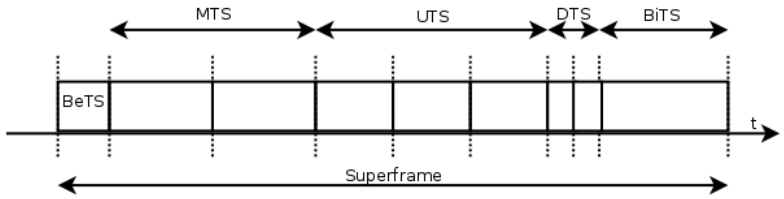
[83], or both as described in [84]; no analysis has been done so far specifically for the LLDN superframe. Instead, a multichannel extension to LLDN has been proposed in [85], which overcomes some scalability issues in a different way.

5.2.2 Enhancement of IEEE 802.15.4.e LLDN

Consider a scenario where every device must be able to send and receive high-priority data as well as low-priority one. If each device has one reserved time slot for uplink and one for downlink, the worst case latency is equal to the superframe duration. A limitation of the LLDN mode is that it offers only reserved uplink slots, while the only downlink slots are BiTS, which are not suited for high-priority data. So, as there is no explicit concept of reserved downlink, the determinism of the communication may suffer, in case of frequent downlink transmissions. To overcome this limitation, Downlink Reserved time slots (DTS) are defined as an optional but relevant component of the superframe. Another limiting aspect for applications with the requirements listed above, is that every UTS and BTS is built over a fixed number of base time slots. This might be flexible enough for most applications, as the unused time for all time slots in a superframe is a small percentage of the superframe duration, however this kind of



(a) Standard IEEE 802.15.4e LLDN superframe format



(b) Modified LLDN superframe format

Figure 5.3: IEEE 802.15.4e LLDN Standard and modified superframe format.

flexibility is not enough when we have the requirements listed above. Moreover, as the BeTS must always fit into at least one BTS, if many uplink packets are shorter than the beacon, a sensible waste of time and bandwidth can occur. The proposed solution tries to optimise the superframe duration, as depicted in Figure 5.3(b), without any optimisation algorithm to compute the BTS length, at the price of a small overhead in the beacon.

Optimised Time Slot Description

The main idea for optimising the superframe is to describe the slot size in a punctual way for every time slot in a superframe. These information will replace the *Timeslot Size* and *Number of Base Timeslots* in the beacon payload. Holding this assumption, there can be two possible approaches:

- adding *Number of TimeSlots* and *Timeslots Durations* fields for UTS, DTS and BiTS and *Timeslots Duration* for BeTS and MTS (since their number is known for the Online State) fields, using 8 bytes in the beacon. This approach is referred to as **BeTStyp**;
- add a field *Timeslots Duration* for each UTS, DTS and BiTS, a field *Number of Data TimeSlots* and two fields *Timeslots Duration* for BeTS and MTS. This will sum to $k_{UTS} + k_{DTS} + k_{BiTS} + 3$ bytes in the beacon. This approach is referred as **BeTSall**.

The first approach works well if traffic requirements are different for uplink and downlink and for high and low priority data, but still has limited flexibility in the case where there are different slot requirements for a same type of slot. The second approach adds a greater overhead in the beacon but provides the flexibility necessary to efficiently handle different classes of traffic.

Time Slot Allocation

With an optimised superframe, the slot allocation algorithm is much simpler since it is not necessary to compute the optimal BTS length, and can be summarised as follows:

1. in the configuration state, the PAN coordinator collects the slot requirements in term of number of bytes to be transferred for each node;

2. when the PAN coordinator has the knowledge of all the time slots that must be available in a superframe, decide a schedule of the time slots simply allocating the time slots in the same order of which they are received in configuration state and sends to each node the duration and offset of the required time slot;

The number of computations required to obtain the schedule will depend solely on the number N of nodes and the number of slots n_i^s required by each node i for a total of $\sum_{i=0}^{N} n_i^s$ operations, with polynomial complexity.

5.2.3 Analytic Results

Throughout this section, a theoretical comparison between standard and modified LLDN is done, regarding worst-case latency. A PHY operating at 2.5 GHZ with O-QPSK modulation, as described in chapter 10 of [57], is assumed. When expressing a time slot length in bytes, the following formula (from 5.2.2.5.2 of [56]) is considered to compute the duration [sec] of the time slot:

$$T = \frac{p \cdot sp + sm \cdot (m + n) + IFS}{v}$$

Where p is the number of octets in the PHY header, m is the number of octets in the MAC header, n is the number of octets in the MAC payload, sp is the number of symbols per octets in PHY header, sm is the number of symbols per octets in PSDU, IFS is the inter-frame space which minimum is $SIFS$ symbols if $m + n \leq 18$ and $LIFS$ symbols if $m + n \geq 18$ and v is the symbol rate (SIFS and LIFS are respectively short and long interframe spaces, described in 8.1.3 of [57] and PHY dependent). For the PHY considered, $p = 6octets, sp = 2sym/octet, sm = 2sym/octet, SIFS = 12sym, LIFS = 40sym, v = 62500sym/s$.
To compare the standard solution with the proposed one, some traffic classes and their requirements have been defined, and superframe structures have been re-elaborated. These requirements concern the number of bytes in MAC payload for every traffic class. The traffic classes that must be considered in a superframe are four:

- uplink high-priority,

- uplink low-priority,

- downlink high-priority,
- downlink low priority.

High-priority traffic shall be sent in dedicated time slots. The conditions imposed, in addition to the ones already specified, are:

1. high-priority traffic in both uplink and downlink for each node, specified as $slot_{up}$ and $slot_{down}$. If there is no ambiguity, both requirements can be specified as $slot_{rt}$,
2. low-priority traffic in both uplink and downlink, specified as $slot_{up\_nonrt} = slot_{down\_nonrt} = slot_{nonrt}$. A total of 2 shared time slots shall be considered, one for uplink and one for downlink,
3. minimal MAC overhead, no security header, no sequence number, so there is 1 byte of mac header and 2 bytes of mac footer,
4. no ACK for high-priority traffic, ack is present for low-priority traffic,
5. no Management Slots in Online State.

Standard Superframe

In order to satisfy the downlink requirements, either a single or double superframe can be considered. In a single superframe, the four traffic classes should be assigned as follows: uplink high-priority to dedicated UTSs, downlink high-priority to dedicated BiTS, low-priority to shared BiTS. In this case, also there can be no consecutive superframes in which BiTSes are set as downlink, to avoid the case in which no low-priority traffic can be sent in the uplink direction. However the worst-case latency for the high-priority downlink traffic doubles with respect to the one for high-priority uplink traffic, which may not be fit in into the requirements of some applications. A different approach is then to consider two superframes, one for uplink traffic and the other for downlink traffic, both composed only of BiTS: part of the BiTSes will be reserved for every device for high-priority traffic and the remaining are shared timeslots for low-priority traffic. This way the worst-case latency becomes the total duration of the two superframes, and is the same for uplink and downlink traffic. Here the BTS size can also be different in the two superframes. A simple algorithm for the search of the optimum BTS length with respect to the superframe length has been implemented performing an exhaustive search, that is:

$$T_{BTS} = \min_{B} T_s$$

$$B = [T_{BeTS}, \ldots, \max(T_{BeTS}, T_{UTS}, T_{DTS}, T_{BiTS})]$$

where T_{BeTS} is calculated according to the standard LLDN specifications and T_{UTS},T_{DTS},T_{BiTS} are calculated according respectively to the $slot_{up}$,$slot_{down}$ and $slot_{nonrt}$ requirements. The total cycle time can then be computed, for the case with a single superframe, as:

$$T_c = T_s = T_{BTS}(1 + N(U_{UTS} + U_{DTS}) + 2U_{BiTS})$$

while, with two superframes, two different T_{BTS}, respectively T_{BTS}^{up} for the uplink superframe T_s^{up} and T_{BTS}^{down} for the downlink superframe T_s^{down} have to be calculated:

$$T_c = T_s^{up} + T_s^{down} = T_{BTS}^{up}(1 + NU_{UTS} + U_{BiTS}) + T_{BTS}^{down}(1 + NU_{DTS} + U_{BiTS})$$

.

Modified Superframe

With an optimised superframe, all requirements can be fulfilled in a single superframe, having all four traffic classes in their corresponding time slot type. The total cycle time can then be computed as follows:

$$T_c = T_s = T_{BeTS} + N(T_{UTS} + T_{DTS}) + 2T_{BiTS}$$

where T_{BeTS} is calculated using either **BeTStyp** or **BeTSall** as described in section 5.2.2.

Discussion

In Figure 5.4, some theoretical results are shown. In Figure 5.4(a) the modified LLDN significantly outperforms standard LLDN with the increase of the number of nodes. The regular steps on standard LLDN are given by the increase of the BTS, since an increase of 8 nodes also increase the beacon length of 1 byte because of the increment in the Group ACK field; the increase ratio is also different. Figure 5.4(b) shows the maximum latency with respect to the required length for one type of reserved time slot, while keeping the other fixed; it is worth noting that the results are the same for UTS and DTS, with the exception of the standard LLDN with 1 superframe, in which case the worst-case latency for downlink traffic is double with respect to the uplink traffic. In this case, Figure 5.4(b) always represents uplink high-priority traffic. When the slot length exceeds 15 bytes, the switch from SIFS to LIFS causes a sudden increase. In Figures 5.4(c) and 5.4(d) the maximum latency with respect to shared slot length is depicted; while in Figure 5.4(c) modified LLDN still performs better than standard LLDN, in figure 5.4(d) the separation between standard and modified LLDN is not clearly defined, and for some values standard LLDN performs the same or better than modified LLDN. This is given by the additional bytes in the beacon slot for modified LLDN. In Table 5.2 some numerical result are shown. The gap between standard and modified LLDN

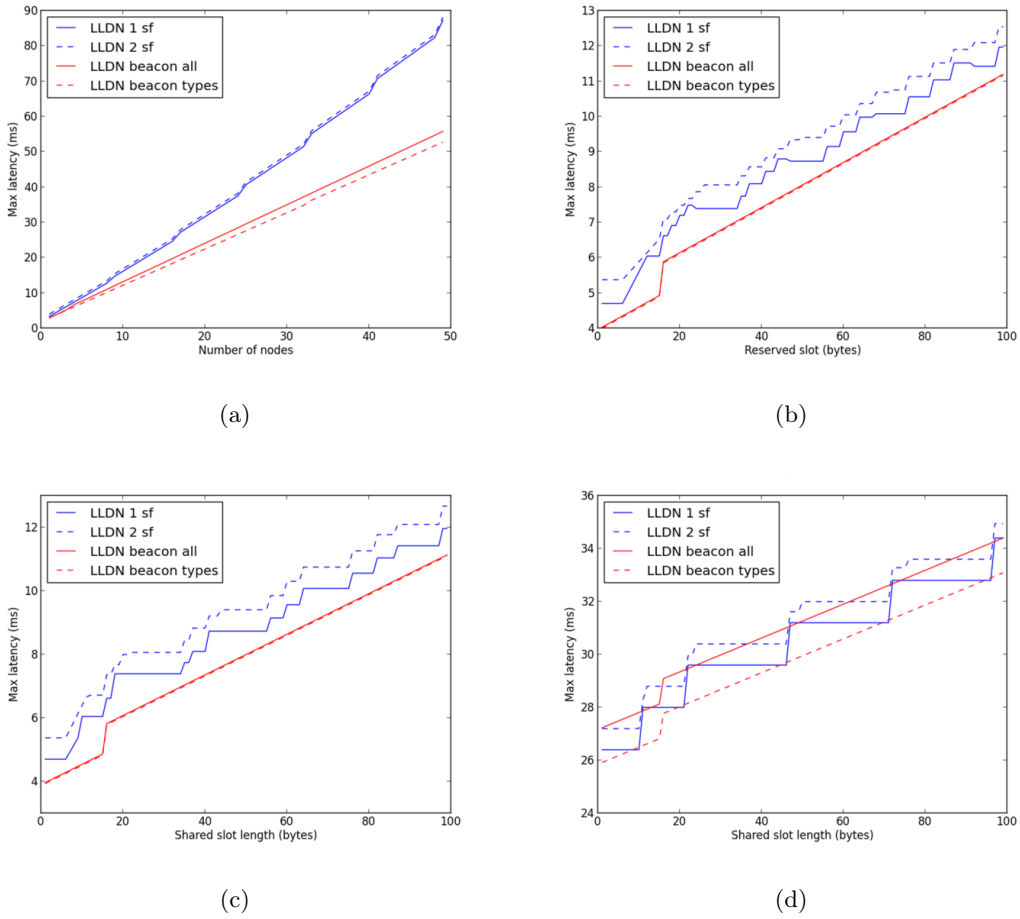


Figure 5.4: Worst-case latency with IEEE 802.15.4e LLDN standard and modified.

is of the order of some milliseconds with varying traffic requirements; this means that given the traffic requirements and the maximum worst-case latency for high-priority traffic, more nodes can be added to the network with modified LLDN. A great improvement is obtained with respect to the number of nodes, with a maximum difference of more than 20 ms with 40 nodes.

5.3 Configuration time analysis and optimisation

In this section the setup process of an IEEE 802.15.4e LLDN network is analysed, first comparing it with a IEEE 802.15.4e DSME configuration (similar to a standard beacon-enabled IEEE 802.15.4), then an optimisation of the configuration process is proposed.

5.3.1 Comparison between IEEE 802.15.4 and 802.15.4e LLDN

The typical superframe for beacon-enabled IEEE 802.15.4 networks consist of a Contention Access Period (CAP) and a Contention Free Period (CFP); while the former is used for asynchronous messages, such as configuration messages, and its access is regulated by a

N	$slot_{rt}$ (bytes)	$slot_{nonrt}$ (bytes)	T_c (ms) STD 1sf	T_c (ms) STD 2sf	T_c (ms) MOD BeTSall	T_c (ms) MOD BeTStyp
10	1	2	16.2	16.8	13.2	12.2
20	1	2	31.6	32.3	24.1	22.5
30	1	2	48.1	49.1	35.0	32.8
40	1	2	66.4	67.2	46.0	43.5
2	10	2	5.6	5.8	4.6	4.6
2	30	2	7.4	8.1	6.7	6.8
2	50	2	8.7	9.4	8.1	8.0
2	70	2	10.0	10.6	9.3	9.3
2	1	10	6.0	6.4	4.6	4.5
2	1	30	7.4	8.1	6.7	6.7
2	1	50	8.7	9.4	8.0	8.001
2	1	70	10.0	10.9	9.3	9.2
15	10	10	26.4	27.2	27.8	26.5
15	10	10	29.6	30.45	30.1	28.9
15	10	30	31.2	32.1	31.3	30.0
15	10	80	32.8	33.6	33.2	32.9

Table 5.2: Numerical results for standard (STD) and modified (MOD) IEEE 802.15.4e LLDN.

slotted CSMA/CA mechanism, the latter is used for high-priority packets, and consists of guaranteed time slots (GTS), where a node can send a packet without using a CSMA/CA mechanism. In IEEE 802.15.4e this operational mode is enhanced with the Deterministic and Synchronous Multi-channel Extension (DSME), while in LLDN mode the network configuration is completely separated from normal operations. In other words, in LLDN the network must be temporarily stopped (no application data can be exchanged) in order to reconfigure the network. However, in many cases this is convenient because, as will be shown in section 5.3.2, configuration time can be in the order of hundreds of milliseconds. This section first focuses on the comparison of two configuration strategies: online configuration, as present in DSME, and offline configuration, as present in LLDN. Then, an optimised Configuration State for LLDN is presented.

IEEE 802.15.4 Slotted CSMA/CA

The configuration of a wireless network involves the discovery of new nodes, as well as the reconfiguration of existing nodes; since the number of active nodes can change during network's

lifetime, there must be the possibility that a new node joins the network; this is possible only if the access scheme used is a multiple access method, since new nodes can't have any reserved time slot. IEEE 802.15.4 defines a slotted CSMA/CA mechanism, designed for time-slotted network, to be used for configuration packets, both in DSME (using MAC commands) and LLDN (using LLDN MAC commands). A CSMA/CA time slot, called in this section also simply time slot, is the portion of superframe reserved for slotted CSMA/CA operations (e.g. a Management Time Slot in LLDN), and a backoff time slot is a fraction of the CSMA/CA time slot, which length is PHY-dependent, and is used as the basic unit of time in the algorithm. The slotted CSMA/CA algorithm as defined in IEEE 802.15.4 has two peculiar characteristics, *backoff deference* and *CCA deference*, as analyzed in [86], and can be summarised as follows:

1. a node wishing to send a packet keeps three different counters, CW, BE, and NB, which first initialises to their default values $CW_0 = 2$, $BE_{min} = 3$ and $NB_0 = 0$ and waits for the next backoff slot to start. All backoff slots, CCA operations and transmissions are aligned with the start of the superframe and the corresponding backoff slot;

2. a delay consisting of a random number of backoff slots is chosen, in the range $[0, \ldots, 2^{BE} - 1]$; the node must then evaluate if the remaining time in the slot is smaller than the backoff delay, in which case it suspends the backoff counter at the end of the current slot, and resumes at the beginning of the next CSMA/CA slot. This is called *backoff deference* mechanism;

3. when the backoff delay expires, the node must evaluate whether there is enough remaining time in the slot to perform CW_0 CCA operations, the transmission of the packet and an optional ACK packet. If this is the case, it starts the first CCA, otherwise it waits for the start of the next CSMA/CA slot, then continues with step 2; this is called *CCA deference* mechanism.

4. if CCA succeeds, it decrements CW, and start next CCA at next backoff slot. If $CW = 0$, the algorithm returns SUCCESS. If CCA fails, it sets $NB = NB+1$, $CW = CW_0$, $BE = min(BE+1, BE_{max})$ and continue with step 2 unless $NB > macMaxCSMABackoffs$, in this case the algorithm returns FAILURE.

The return value of the algorithm is reported to the MAC layer; in case of SUCCESS, the MAC layer shall start the transmission of the packet, otherwise it may restart the CSMA/CA algorithm if retransmissions are enabled. However there is some critical point to consider when using IEEE 802.15.4 slotted CSMA/CA algorithm: first of all, the backoff slot duration should be as close as possible to the maximum propagation delay, as showed in [87] and [88], and also an appropriate CSMA/CA time slot length should be used. A short time slot means that

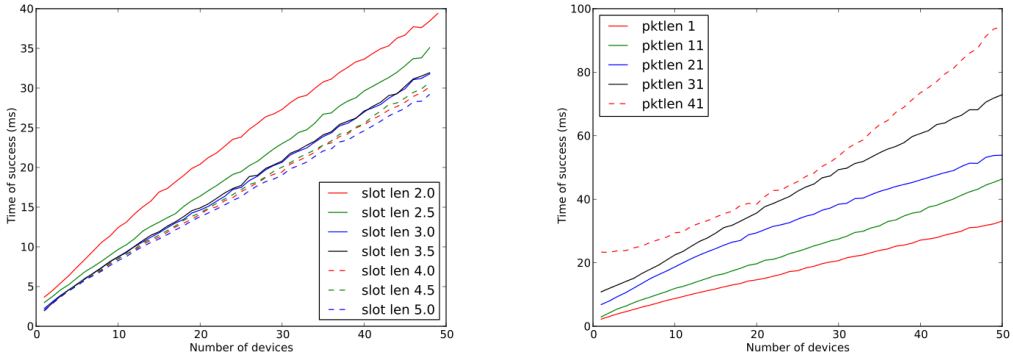
the resulting superframe will be shorter, hence reducing the cycle time, but may result in a poor efficiency of the slotted CSMA/CA algorithm, because of CCA deference. On the other hand, a long time slot will result in a longer superframe, but will increase the efficiency of the CSMA/CA, reducing the access time. Moreover, in [86] the authors show that the deference mechanisms, although reducing collision probability, have a direct impact on transmission delay. A complete theoretical model for 802.15.4 slotted CSMA/CA is, as far as the authors know, not yet fully developed, although there are several examples of partial models, such as [89] but many consider a different CCA deference mechanism, which was changed starting with IEEE 802.15.4-2006, or only the backoff deference, for example [90]. While using only backoff deference has the useful effect of simplifying the modelling of slotted CSMA/CA (since it hides the effect of the start or end of a CSMA/CA slot during backoff), it is reasonable to think that this is not the case for CCA deference, since this mechanism adds the possibility that after the end of the backoff, instead of performing a CCA, the algorithm goes back to backoff state. The aim here is not to develop a new model for CSMA/CA, rather to compare configuration strategies that employ CSMA/CA as access method, so the analysis has been performed using a discrete-events network simulator.

Simulation results All simulations results presented in this thesis have been obtained using the LR-WPAN module of NS3 v3.21[1] network simulator with additional back-ported bug fixes and a custom MAC layer on top of a 2.4 GHz O-QPSK PHY. The channel has been characterised using a three log-distance propagation loss model and constant speed propagation delay model; for each iteration nodes are randomly placed on a 100x100m grid.

IEEE 802.15.4 Slotted CSMA/CA For the purpose of characterising the time needed for discovery and configuration of N nodes in a WSN, it is interesting to see the impact of each parameter on the average time needed for a successful transmission. It is worth noting that this is different than considering the access time, i.e. the time needed for one or more CSMA/CA execution to return SUCCESS, since the effect of collisions can not be neglected due to the great length of the backoff slot time, as showed in section 5.3.1. To this end, simulations have been performed measuring the average time for a successful transmission depending on three parameters: the length of the CSMA/CA slot, the number of nodes contending the channel access, and the length of the packet. The transmission time is measured considering only the time spent in the CSMA/CA slot; as such, this would be the real transmission time if the superframe were composed only by one such slot.

In Figure 5.5(a) and 5.5(b) the average transmission delay is reported, depending on the

[1] https://www.nsnam.org/release/ns-allinone-3.21.tar.bz2

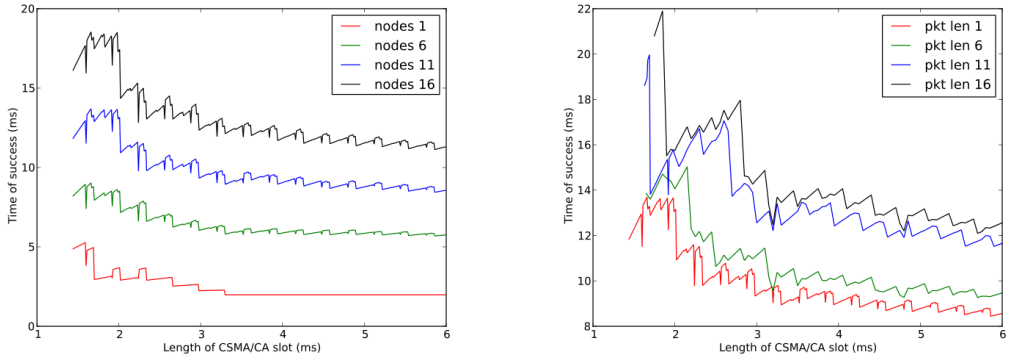


(a) (b)

Figure 5.5: Average time for successful transmission with slotted CSMA/CA, depending on the number of nodes. In 5.5(a) packet length is fixed to 1 byte; in 5.5(b) CSMA/CA slot length is fixed to 3 ms.

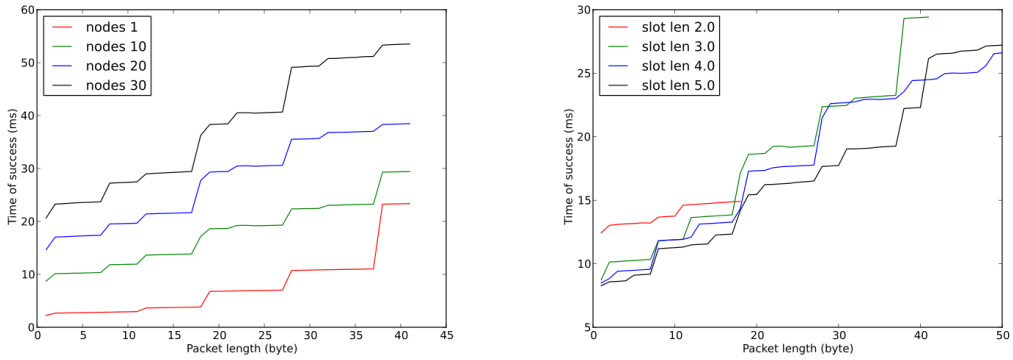
number of nodes. Figure 5.5(a) shows that if the CSMA slot is sufficiently short, the delay increases with high rate with respect to the number of nodes in a non linear way, then it settles to a linear growth. As reported in section 5.3.2, this can be explained with the fact that if the number of nodes is below a certain threshold, depending on the number of nodes and on the length of the packet, the probability to have a node attempting to transmit is low, and the time slot is not fully used for channel contention, but all nodes are in backoff state. Increasing the number of nodes, this percentage of time shrinks until for the whole duration of the CSMA/CA slot there is at least one node contending the channel; from this threshold on, the average time increases linearly. This can be explained as an effect of the slotted nature of CSMA and in particular of the CCA deference mechanism, since this behaviour is less visible if the CSMA slot is sufficiently long, when the CCA deference happens with lower probability, and the algorithm eventually degenerate into an unslotted CSMA/CA. Figure 5.5(b) shows that for low packet sizes, the growth of transmission time is almost linear, but for big packets, which transmission time added to the CCAs time approaches the CSMA/CA time slot length, this evolves in a more than linear growth. This shows the importance of using short packets, in order to reduce the average transmission time in slotted CSMA/CA.

In 5.6(a) and 5.6(b) the average transmission delay is reported with regard to the length of the CSMA/CA slot. From 5.6(a) can be seen that if the slot length remains under a certain threshold, there is a strong dependency of the transmission delay on the slot length, but as soon as this threshold is exceeded, increasing the slot length has little effect on the average transmission delay. This suggests that there is little advantage in having a long CSMA/CA time slot, after a given threshold. Similarly, in 5.6(b), the greater variations of transmission time are located where the CSMA/CA slot length is small. It is worth noting that, increasing



(a) (b)

Figure 5.6: Average time for successful transmission with slotted CSMA/CA, depending on CSMA/CA slot length. In 5.6(a) packet length is fixed to 1 byte; in 5.6(b) the number of nodes is fixed to 11.



(a) (b)

Figure 5.7: Average time for successful transmission with slotted CSMA/CA, depending on packet length. In 5.7(a) CSMA/CA slot length is fixed to 3 ms; in 5.7(b) the number of nodes is fixed to 11.

the CSMA/CA slot time, there is a saw-like behaviour of transmission time; this can be explained considering that every operation in slotted CSMA/CA must take place at backoff slot boundary, so if the CSMA/CA slot is increased of an amount equal to a backoff slot (20 symbols, 320us for 2.4 GHz O-QPSK), there is the possibility that a greater backoff period can be contained in the same superframe. The oscillation are then given by the long backoff time slot.

In Figures 5.7(a) and 5.7(b) the average transmission delay is reported with regard to the length in bytes of the packet to be transmitted. A stair-like behaviour is visible in both figures; this is related to the fact that increasing the packet length of an amount such that the transmission time at PHY level increases of an amount that equals the backoff slot, the time remaining in a CSMA/CA slot for the backoff state decreases, and this is also related to

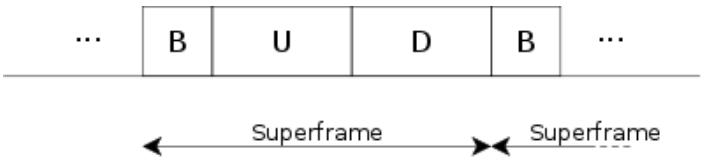


Figure 5.8: Superframe structure for Discovery and Configuration phase for the Offline configuration strategy. B is the beacon slot, U is the uplink slot, D is the downlink slot.

the fact that every operation in slotted CSMA/CA must take place at backoff slot boundary. Comparing 5.7(a) and 5.7(b), it can be noted that the slot length has little influence at varying packet length, compared to the number of nodes.

Configuration performances

Offline Configuration Description In this scenario the discovery of the nodes and the configuration of the network requires a dedicated superframe, as shown in Figure 5.8, so the discovery and configuration phases are mutually exclusive with the online phase, where application data can be exchanged by the nodes. This is the case for LLDN mode in 802.15.4e, where Discovery and Configurations states have a superframe composed of three time slots: one reserved for the beacon packet, one for downlink packets and one for uplink packets. For both uplink and downlink slots, in addition to slotted CSMA/CA, an RTS/CTS procedure is needed, as in 802.15.4e, since the collision probability is supposed to be high, given that every node in the network is competing for the channel access. It is worth noting that in uplink slot there are only device nodes competing for channel access, while in the downlink slot there is the PAN coordinator alone. A theoretical analysis would then need to combine these two traffic conditions. The Discovery phase expects each device to send a Discovery Response packet, in order to advertise its presence to PAN coordinator, and the PAN coordinator to send back a Discovery Response Acknowledge packet to each node. When all Discovery Response Acknowledge packets are sent, or a given maximum amount of time is elapsed, the network switches to Configuration state. Each node sends a Configuration Status packet, containing the last configuration of the node, if any, and the PAN coordinator answers with a Configuration Request packet, containing the new configuration for each node. Each node must then accept the new configuration, sending a Configuration Acknowledge packet. This procedure is illustrated in Figure 34c of IEEE 802.15.4e.

Online Configuration Description In this scenario the discovery and configuration of the nodes are performed in the same superframe that is used to exchange application data. This is the case of the beacon-enabled modes other than LLDN in IEEE 802.15.4 and 802.15.4e.

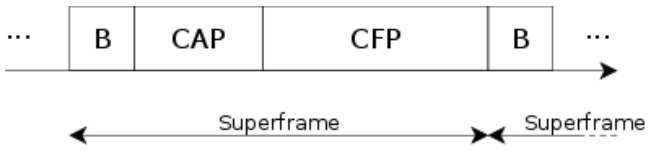


Figure 5.9: Superframe structure for the Online configuration strategy. B is the beacon slot, CAP is the Contention Access Period slot, CFP is the Contention Free Period slot.

Here the superframe is divided into three main periods: beacon slot, Contention Access Period (CAP) and Contention Free Period (CFP), plus an optional idle period. Moreover, the duration of each of these periods can not be arbitrarily chosen, but is dependent on the Beacon Order (BO) and Superframe Order (SO) and the number of nodes which have a Guaranteed Time Slot (GTS) assigned in the CFP. The communication in the CAP, as opposed to LLDN Discovery and Configuration, is bidirectional, that is the same time slot is used for both downlink and uplink packets. This has the effect that for the same number of nodes, slot length and packet length, the time needed for the configuration of a number N of nodes, considering only the CFP, is greater than in the Offline scenario. In standard IEEE 802.15.4 beacon-enabled mode, the steps required for a device in order to reserve a GTS are association and GTS request. The association starts with the device sending an Association Request, then the device must poll the PAN coordinator with a Data Request packet, in order to receive an Association Response. These three packets require an explicit Acknowledge packet. Once associated, the device sends a GTS Request packet, and wait for the response which will be embedded in the beacon packet by the PAN coordinator.

Simulation Setup In order to compare the Offline and Online Configuration with regard to the superframe composition, the same traffic pattern must be used. Here, it has been chosen to use the traffic pattern described in section 5.3.1, that is two packets for the discovery of the nodes and three packets for the configuration of each node, using a separate acknowledge. Furthermore, no RTS/CTS mechanism is considered. The comparison has been carried out considering the average setup time with respect to the number of nodes in the network and the CSMA/CA time slot length. The length of the relevant packets is reported in Table 5.3. All Acknowledge packets used in the simulations have as payload the Extended MAC Address of the device node which sends or receives the packet.

Offline Configuration Results In Figure 5.10(a) the average time required for the completion of the sequence composed by Discovery and Configuration phase is reported, depending on the number of nodes. It is interesting to note that the increment is approximately linear. In Figure 5.10(b) there is the same behaviour as in Figure 5.6(a) for the first part of the curve,

(a) Discovery Response packet.

Bytes	Description
1	MAC CMD identifier
8	Extended MAC address
12	Requested Slot Description

(b) Acknowledge packet.

Bytes	Description
1	ACK identifier
0..n	Payload (optional)

(c) Configuration Status packet.

Bytes	Description
1	MAC CMD identifier
1	Configuration ID
2	Short MAC address
8	Extended MAC address
12	Requested Slot Description

(d) Configuration Request packet.

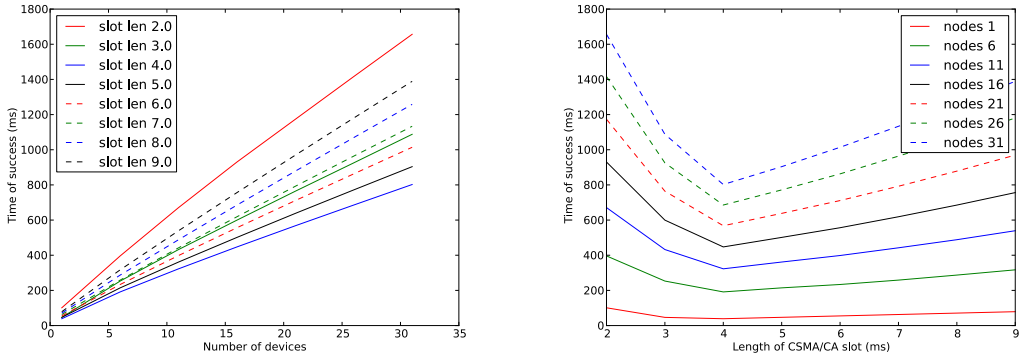
Bytes	Description
1	MAC CMD identifier
1	Configuration ID
2	Short MAC address
8	Extended MAC address
12	Assigned Slot Description

Table 5.3: Detailed format for packets used in LLDN Discovery and Configuration State.

but in this case after the threshold there is an inversion of the slope of the curve, which can be explained with the fact that for this configuration pattern, the increase in length of the CSMA/CA slot has effect both for the uplink and downlink slot, so for long CSMA/CA slots the performance decrease, as only the downlink slot benefits from this increase. This means that there is an optimal value for the CSMA/CA slot length, and it seems to be independent from the number of nodes which must be configured.

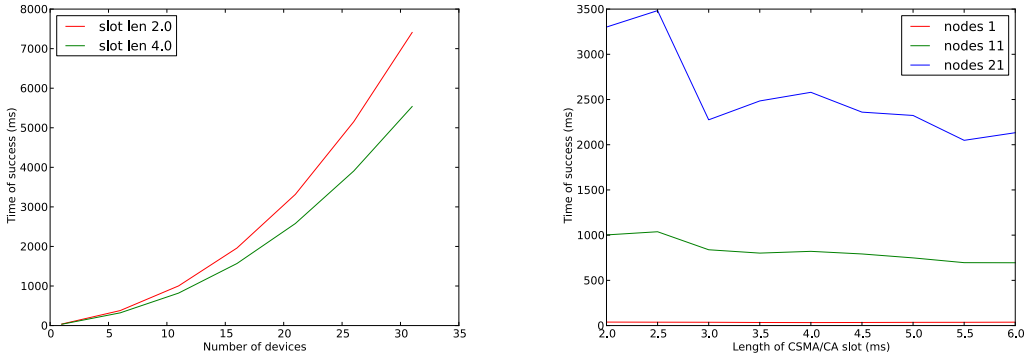
Online Configuration Results In Figure 5.11(a), the average time required for the configuration is reported, depending on the number of nodes and assuming a constant superframe, that there is no GTS allocated. The increase of configuration time seems polynomial, as opposed to Figure 5.10(a). Furthermore, for the same number of nodes, the configuration time is much bigger, and this can be explained with the fact that downlink packets here are sent in the same CSMA/CA slot as uplink packets. This on one hand contributes to the average time needed for the transmission of a packet, on the other hand delays the whole configuration process. In Figure 5.11(b) a behaviour similar to Figure 5.6(a) is obtained, that is for small slot lengths the average configuration time decreases, and it settles for sufficiently long time slot lengths.

Discussion The comparison of Figures 5.10(a), 5.10(b) with 5.11(a), 5.11(b) clearly express the advantages and disadvantages of both Offline and Online Configuration. While the former allows for rapid reconfiguration, it temporarily blocks the whole network. The latter does not need to block the application traffic to be exchanged, but it requires an amount of time



(a) (b)

Figure 5.10: Average time for the completion of Offline Configuration, depending 5.10(a) on the number of nodes and 5.10(b) on the CSMA/CA time slot length



(a) (b)

Figure 5.11: Average time for the completion of Online Configuration, depending 5.11(a) on the number of nodes and 5.11(b) on the CSMA/CA time slot length

much greater than Offline Configuration. The temporary block of the network may however be more admissible for an application that requires frequent re-configurations of the network. For Offline configuration however it is crucial to use a CSMA/CA slot length near the optimum value, as shown in Figure 5.10(b), in order to achieve the maximum performance. This issue is not present in Online configuration, as shown in Figure 5.11(b).

5.3.2 Enhancement over IEEE 802.15.4.e LLDN

The configuration state, as described in 5.1.2 and showed in 5.3, has the disadvantage of having a non-deterministic duration. Provided that in Configuration State each node has to go through the fixed Configuration process (i.e. Configuration Status/Request/ACK) and the PANc already knows which devices are present in the network, the different approach considered here is to reduce the time required for the configuration process using Reserved

Management Timeslot (RMTS) instead of standard MTS, which are shared among all nodes. As results will show, the time required to complete the Configuration process results:

- deterministic and linearly dependent solely on the number of devices involved;
- up to six fold faster.

Assuming a RMTS for each device, no hidden node in the CSMA/CA, and an ideal communication channel (i.e. error-less and with null propagation delay), the total configuration time can be computed as two superframe cycles; in the first one, each node sends the Configuration Status packet, while, in the second one, the PANc sends the Configuration Request to each device and receive the corresponding ACK, as described at sec. 5.1.9.3 of [56]. Each device can obtain its RTMS either explicitly (i.e. in the ACK payload in Discovery State) or implicitly through an algorithm based on its MAC address and some parameters included in the beacon. An approach like the one adopted in [91] could be used as a starting point.

Simulation results

To compare the proposed configuration scheme with the standard LLDN configuration scheme, one has to consider that all slots are composed by BTSs, as imposed by LLDN mode. The slot length will be equal to the length required by the greater packet among the beacon, the Configuration Status, the Configuration Request and the ACK. As IEEE 802.15.4e amendment does not specify the format of the packets Configuration Status and Request, they will be defined here, to compare the standard approach with the proposed one, as follows:

- 1-octet with the Command Frame ID;
- 1-octet with configuration sequence number;
- 8-octets with the extended MAC address;
- 2-octets with the short MAC address;
- 1-octet with the type of the required slot in Configuration Status packet, and the indication of the channel used in Configuration Request packet;
- 1-octet with the length of the required slot in Configuration Status packet, and the number of MTS for Online State in Configuration Request packet;
- 1-octet with the type of the assigned slot;
- 1-octet with the length of the assigned slot.

The comparison assumes that, at the beginning of the Configuration State, a device has no slots assigned and it is requesting one UTS to the PAN coordinator. A minimal LLDN MAC Command frame header is also assumed.

Standard Configuration Rather than developing a variant of the Markov Chain model suited for the combination of CSMA/CA and the RTS/CTS procedure, numerical results are obtained through a statistical characterization of the configuration time by means of Monte Carlo simulation, based on the specifications of the IEEE 802.15.4e amendment [56]. Management Timeslots are assumed to be equal to 7 BTSs, which is the maximum allowed by LLDN.

Modified Configuration For this scheme, the overall configuration time can be computed as:

$$T_{config} = 2T_{BTS}(1 + 2N)$$

where T_{config} is configuration cycle time and N is the number of devices. It is assumed that each device obtains its RMTS in Discovery State, so no additional parameters must be added to the beacon in Configuration State.

Discussion Figure 5.12 plots the overall time required for the configuration of N nodes, both for standard and modified LLDN. The modified LLDN performs significantly better than standard LLDN, in terms of determinism and Configuration State duration, because there is no CSMA/CA involved. It is interesting to note that, for standard LLDN, the configuration time initially grows rapidly, but it settles down to a linear trend after a certain number of nodes are considered. This can be motivated by the stochastic nature of the backoff time in the modified CSMA/CA algorithm: if the number of nodes is relatively small, the probability that a shared slot is used to transmit its maximum allowable number of packets (considering the Clear Channel Assessments and the RTS/CTS packets) is also small, because the number of backoff slots actually used will cover the whole range of available backoff slots with low probability. Increasing the number of nodes, this probability increases and asymptotically tends to 1. Such behaviour is confirmed by simulations with up to 200 nodes.

5.4 Safety application protocol

At the application layer the general requirement given by safety standards is to have a certain diagnostic coverage, that is the ability to detect errors is a quantifiable way, e.g. as a percentage of the total possible errors or a residual probability error. Real-time issues only represent a fraction of the possible errors; in addition to the classic transmission error given by noise on

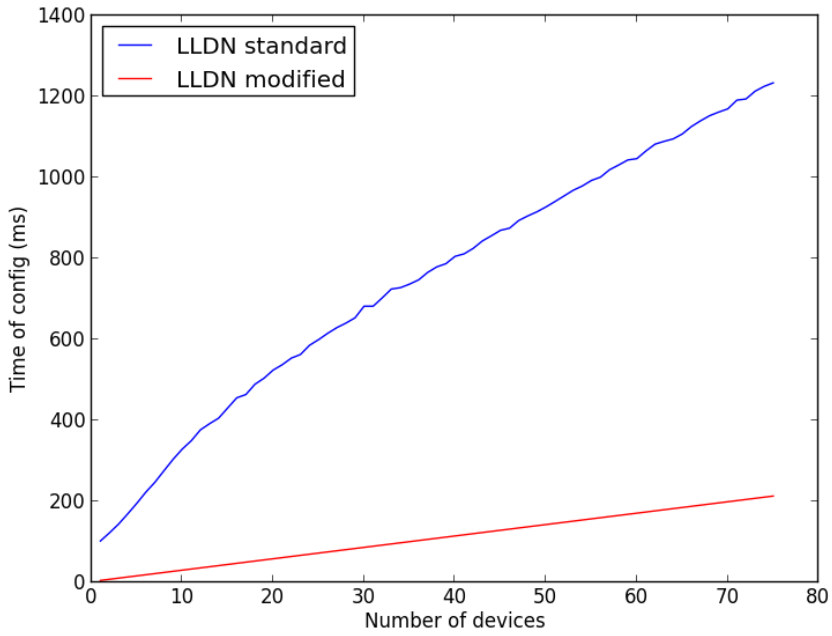


Figure 5.12: Configuration time plot, highlighting the time taken for procedure adhering to the standard (Monte Carlo simulations) and with the proposed modified approach (deterministic)

the communication channel, other errors can derive from the interference of other ECU on the same communication network, from a characteristic of the network itself (e.g. incorrect sequence, possible for example on IP networks), from a maliciously injected packet or from a software error (e.g. the infamous babbling idiot). The strategies to detect this kind of errors are listed in relevant safety standards like ISO 15998 and ISO 25119, and are also applied in industrial products like openSAFETY [54] in industrial wired application domains. Here a safety layer suitable for WSN is presented. With reference to Table 2.1 the chosen countermeasures are *Running Number*, *Timeout*, *Sender/Receiver Identifier*, *Check of Data Consistency*. An additional header is then inserted in the data packets, and as a further optimisation the timing information are expressed in a specific domain on the WSN. The PAN coordinator, which usually is the gateway to external networks and data collector, translates the timing information from the WSN domain (with relative timestamp and moving reference) to the external time domain (e.g. UTC timestamp relative to 00:00:00 1/1/1970). The safety headers in both cases are visible in Figures 5.13(a) and 5.13(b). In order to create a separate time domain, with shorter timestamp and moving references, an additional data payload is added to the periodic beacon transmitted from the PAN coordinator. This is explicitly allowed by the IEEE 802.15.4 specifications. This additional beacon payload is visible in Figure 5.13(c) and contains two time references, to securely handle the overflow of the counter. The second time reference shall have a delay of $\frac{T_p}{2}$ from the first reference, where T_p if the WSN time

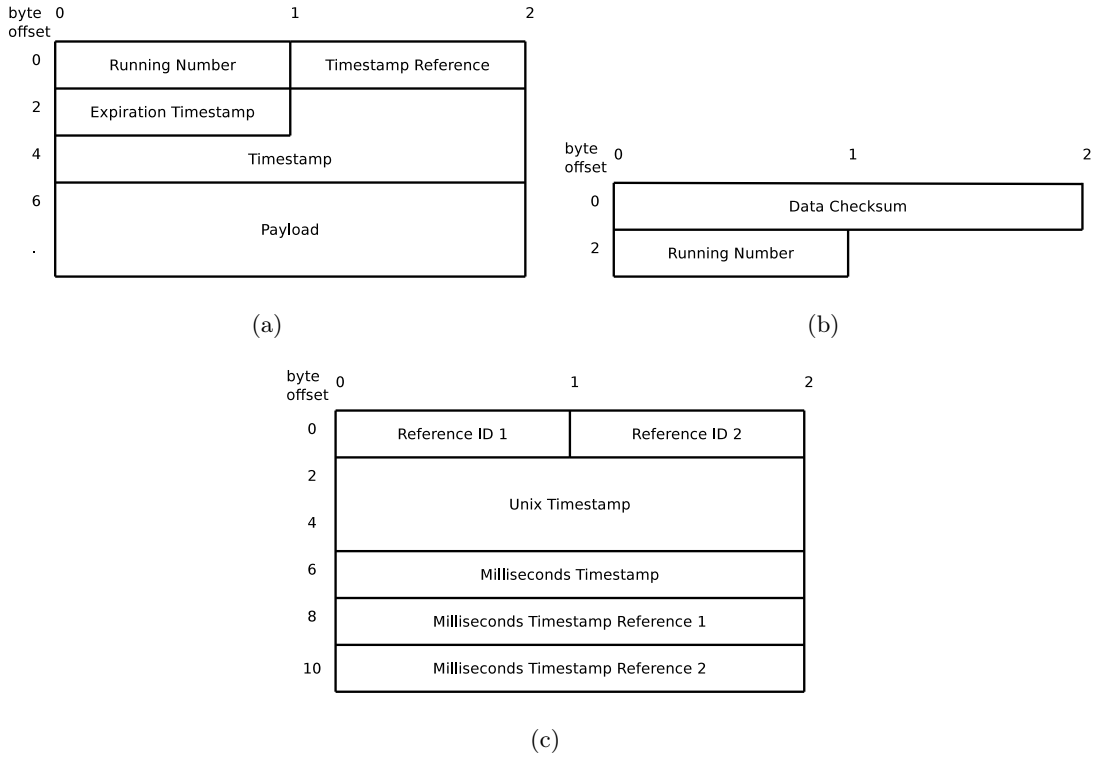


Figure 5.13: Detailed format of 5.13(a) safety protocol header, 5.13(b) safety acknowledge and 5.13(c) beacon payload used for time synchronisation on WSN.

period, so that they will overflow at different moments in time, and the sensor is able to choose the appropriate time reference. An additional safety margin can be applied on the timestamp values, restricting the validity range of a timestamp to $[0.1T_p \ldots 0.9T_p]$.

5.5 Experimental implementation

An experimental implementation of the modified IEEE 802.15.4e LLDN protocol has been realised at CNR-IMAMOTER within the FACTOTHUMS project. Unfortunately at the time the available COTS hardware for WSN (Zolertia Z1) were not suited for precise timing as required by LLDN with the reduced superframe optimisation. This is mainly due to the fact that COTS module based on msp430 use the internal RC oscillator to save power, with the disadvantage of having a 2-3% jitter on the clock frequency. This makes an experimental validation of the modified LLDN almost impossible, since such variation forced to increase the timeslot duration in order to receive a given packet in the intended time slot, making the optimisation impact impossible to asses.

This is also one of the reasons to develop a new hardware platform; at the time of writing (2016) there are ARM Cortex M SoC with an embedded IEEE 802.15.4 radio transceiver, which can reach a higher clock frequency and the available modules have a crystal oscillator. Further detail are given in section 6.4.

Chapter 6

Advanced Architectures for WSN

This chapter presents the hardware and software architectures for a safe WSN. At the hardware level, different safe architectures are possible, all applicable to a safe WSN node, depending on specific needs or constraints. While these architectures are well known from functional safety standards and other application domains, its applicability to a WSN node has not been considered in the literature and is discussed here. At the software level, an overview of the requirements of relevant functional safety standards is given, then some WSN operating systems are analysed with respect to these requirements and evaluated for suitability. The goal is to assess the current status of common WSN software in order to allow a development plan to be drawn, reusing existing software where possible; however a complete software development plan is not reported here. Finally, at the end of the chapter a prototype implementation is presented, developed at CNR-IMAMOTER.

6.1 Hardware Safety Architectures

6.1.1 Safe architectures for WSN

Developing the hardware architecture of the WSN node, ISO 25119 was chosen as a main reference standard because is the youngest among functional safety standards and the most complete, especially with regard to software, even if specific for Agricultural machines. This chapter proposes an effective solution for fail-safe systems which include wireless sensor nodes. In fact, as described in [92], many possible architectures are available, but, as shown in [93] and in [94], a double microcontroller architecture is the most used and well proved architecture for fail safe X-by-wire systems. Even if they are different from WSN nodes, the microcontroller reliability and its monitoring present the same critical aspects. An assumption of this chapter is that in WSN nodes the safety goal is to detect the system malfunction and to switch off damaged nodes in order to preserve the safety of the communication channel; a fail silent

system shall then be designed.

6.1.2 Safety Considerations

The main regulations to which it is possible to refer for hardware and software design for industrial application are the ISO 13849 and the IEC 62061, depending on the complexity of the electronic systems. WSN are simple systems and the ISO 13849 is sufficient for component and system analysis. Both these standards are mainly focused on system and hardware specifications, while the software quality is not very well analysed; for that reason also the ISO 25119 is used. ISO 13849 and ISO 25119 are very similar for hardware categorisation; for the sake of simplicity in this chapter the notation of the ISO 25119 is adopted, as defined in section 2.2.1.

6.1.3 State of the art

A single WSN node, seen as a typical elaboration system, is composed of an input block, reading the input data (the sensor data acquisition) which are elaborated by a logic part (a microcontroller) and are sent to the output block (the radio transceiver), as seen in Figure 6.1. It can be seen that this correspond to an hardware category 1 of Figure 2.2. The typical sensor node is then a single channel system, thus the hardware architectures used in industrial applications to guarantee a high reliability can be applied to a WSN node and the system using it. A WSN node is normally composed by a low-power microcontroller, for example based on the ARM architecture, the 8051 architecture or on some extremely low-power architecture such as MSP430, which reads some sensors through its analogue or digital inputs, and a radio transceiver which is used to transmit the data on the WSN, usually to a network coordinator or gateway. The microcontroller and the transceiver can in some cases be embedded in the same chip, for example as in the TI CC2538 System on Chip [95]; in this case any action performed on the microcontroller (e.g. reset or power) can directly affect the transceiver. WSNs are often built using COTS modules rather than custom hardware; for example the Bluegiga BLE112 [96] or the Zolertia Z1 [97]. An example of a standalone transceiver is the Texas Instruments CC2520 [98]. Normally the microcontroller reads a single sensor for each data which has to be measured, and the conditioning circuit is as simple as possible, in order to save as much power as possible. All the commercial modules have only one microcontroller, for both cost and power consumption reasons. The reliability achievable with a single microcontroller is usually sufficient for a typical non-safety relevant application.

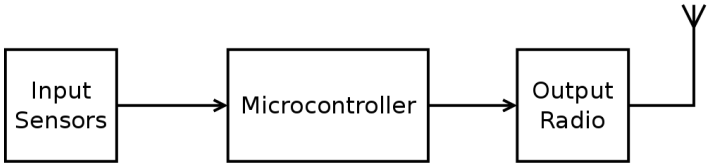


Figure 6.1: WSN node typical architecture

6.1.4 Safety requirements for WSN nodes

As seen in chapter 2, a certain Functional Safety Performance Level can be achieved with a combination of hardware architecture category, component quality, software required level and diagnostic coverage. While a single-channel architecture is sufficient for the lower safety levels, at the price of a higher software required level, it is better to choose a hardware category 2 and have a less stringent requirement for software. This architecture is chosen for two reasons: first, it is a future-proof architecture, which can be used in many different context, allowing to reach higher performance levels; second, it can reduce the time and effort spent in software development because it allows for a lower SRL, eventually allowing the reuse of existing legacy code as a starting point for low and medium functional safety performance levels. The most noticeable difference between the single channel architecture (category 1) and the single channel with test equipment (category 2) is that, if a timeout is included in the safety measures of the WSN gateway, the single fault of a node can't lead to the loss of the WSN safety function, because in case of any recognised unrecoverable error the test equipment will not allow the transmission of wrong data. This can't be achieved in case of single channel where, as an example, the simple "babbling idiot" malfunction could affect the entire WSN data transmission, as seen in Figure 6.2.

6.1.5 Hardware architecture for WSN nodes

In order to underline the main differences in error handling capability of the different solutions, the discussion will consider different diagnostic coverage levels. To fulfil these requirements it is necessary that the MMC is able to detect errors in the input and that the SMC is able to detect errors in both the MMC and the network communication handling. These aspects determine the reliability of the data received through the WSN from the gateway.

As stated before, the node reliability depends on two main factors: the integrity of the transducer, in particular the ability of the MMC to be aware of transducer's malfunctions, and the correctness of the elaboration carried out by the MMC on the acquired data. Moreover, the SMC has to be able to diagnose the transceiver faults, and to react accordingly.

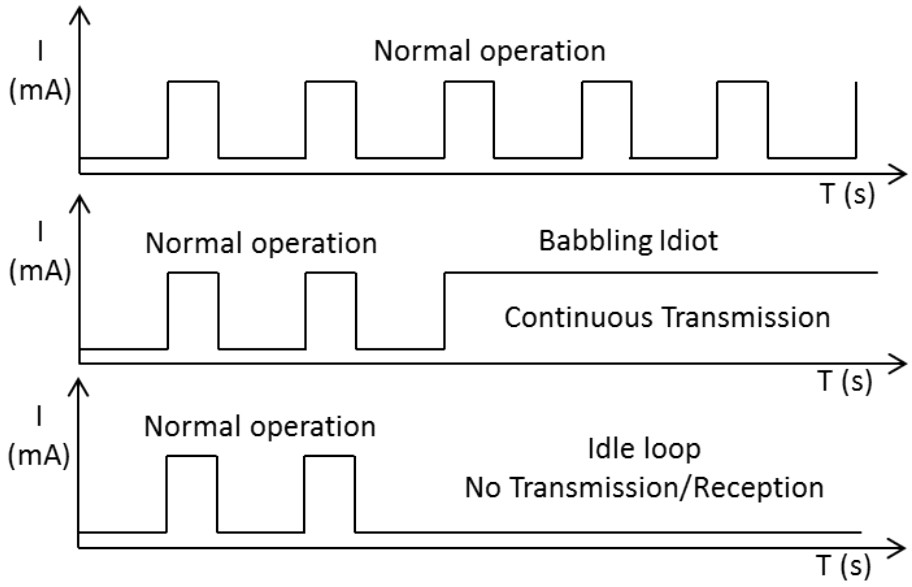


Figure 6.2: Power consumption during normal operation compared to consumption during transceiver problems

Input Block

To be able to reveal an error in the input of the main channel there are mainly two ways. It is possible to use diagnosable transducers, for example input sensors which signal is in the $[0,5 \ldots 4,5]$ V range; these are particular transducers where the value read corresponds through a characteristic curve to the physical value and that will always be inside the $[0,5 \ldots 4,5]$ V range, which is smaller than the one between positive and negative power supply range. This ensures the ability to reveal "short to power supply" and "short to ground" errors. Furthermore, if the physical value to be read has a known variation range it is possible to reveal an error when the value is outside that range for a reasonable time. However it is not possible to reveal neither to correct a drift in the acquired value.

Another strategy that can be adopted is to apply the redundancy principle that consists in using more than one transducer to acquire the same value. In this situation it is a good practice to apply also the diversity principle using transducers with different characteristics or from different manufacturers to avoid systematic errors. If the solution adopted is to use more than one transducer, it is necessary to decide how many of them are needed, having a trade-off between reliability, costs and performances (power consumption, latency, etc.). If it is chosen to use only two transducers it is possible to reveal drift errors, in addition to "short to ground" and "short to power supply", by monitoring the difference between the physical values read by the two sensors. If it is significant, an error will be revealed, but it is not possible to apply any error correction strategy. When using three or more transducers the cost and the power consumption increases but it will be possible both to reveal an error and to correct it. The

error revelation strategy is the same stated before, but with three transducers it is possible to identify the sensor that is in error and to continue to work by using the remaining ones.
To comply with an hardware category 2 and the principle of simply revealing faults and non-reliable information, it is enough to be able to reveal an error. In consequence of that a good compromise can be to use a double transducer that can grant the coverage of both short circuit errors and drift errors.

Logic Block

The following element in the main channel is the microcontroller that has to acquire the information from the transducer and to forward it to the transceiver. To comply with hardware category 2 it is necessary that a MMC error is revealed by the SMC and that this supervisor can act on the main microcontroller and on the output to avoid erroneous conditions, in this case to avoid the transmission of wrong data. The SMC will communicate with the MMC by using a *test channel*, which is usually an in-circuit bus like I2C or SPI. This communication is necessary for the safety microcontroller to continuously test the correctness of the functions of the main microcontroller. These tests consist of a smart watchdog testing in which the safety microcontroller expects to receive a message periodically, an ALU test by giving to the MMC a random number that will be elaborated through a series of fixed operation and the result will be sent back to the SMC. Even the acquisition system of the MMC can be tested. For example, if the transducer is acquired through an analogue port it can be tested by letting the MMC acquire a shared input or an special input configurable by the SMC, and sharing through the test channel the read value.

Output Block

The last element of the chain is the wireless transceiver, which must be diagnosed by the SMC. For this purpose it is possible to use a simple but very efficient solution: to monitor, through a shunt resistor or other hardware circuits, the current consumption of the transceiver. This is possible thanks to the remarkable difference that exists between the current consumption during transmission or reception compared to the consumption during idle time of the communication, where the transceiver is usually turned off. If an high power consumption is revealed for a long period of time it means that the transceiver is experiencing some kind of problem, for example it can be blocked in transmission or in reception. In fact, usually the transceiver is switched on only when it has to transmit or when it is likely to receive data, and it does not stay powered on for a long period of time. An example of power consumption during normal working phase compared to that of a faulty behaviour is shown in Figure 6.2. When problems on the transceiver are detected the SMC could try to reset the transceiver through reset signal

(if present) or by acting on its power supply.

6.1.6 Example on existing WSN modules

None of the commercial modules currently available satisfy the requirements listed in previous sections. For this reason, three alternative versions of an hardware category 2 architecture have been identified, which are composed of standard commercially available components:

- Transceiver-only module, with independent MMC and SMC.
- Transceiver module with on-chip but separated MMC and independent SMC.
- MMC with integrated transceiver and independent SMC.

Each case represent a baseline upon which further modifications are possible to comply with hardware requirements. The most flexible case is the transceiver-only module. In this situation the carrier board will include the MMC, the SMC and the sensor. In this situation it is possible to choose every component basing on the application needs. It is possible to choose how many transducers will compose the input part and the most appropriate microcontrollers. Moreover both MMC and SMC will have direct access to the transceiver granting the ability to perform the necessary actions to reach the safe state, that is to stop the wireless communication. In the second scenario the COTS module has an on-board MMC, then the carrier board will have to include the input part and the SMC. In this situation the SMC has often a reduced ability to perform any action on the transceiver, because it may only be able to interact with the main microcontroller, and not have direct access to the transceiver. In case of fault, the only way to disable the communication is by switching off the power supply of the whole module (for example through a transistor). For the last case, in which the transceiver is integrated with the main microcontroller, the same considerations apply, but it must be noted that in this case the transceiver is certainly not reachable from the SMC, having a strong interdependence with the MMC. The described architectures are shown in Figure 6.3. It can be noted that in the figure the related components of the category 2 architecture are identified in the three possible WSN node hardware configurations, adding to the classic scheme only the power supply as it is the method to independently control the main output for the SMC.

6.2 Software Safe Architectures

On embedded systems there is no single method to build safety-critical software. Each application and each company has its methods and conventions so, from a standards point of view, the only thing that can be done is to evaluate some degree of software quality to represent the probability of residual error. It is worth noting that software bugs are classified as

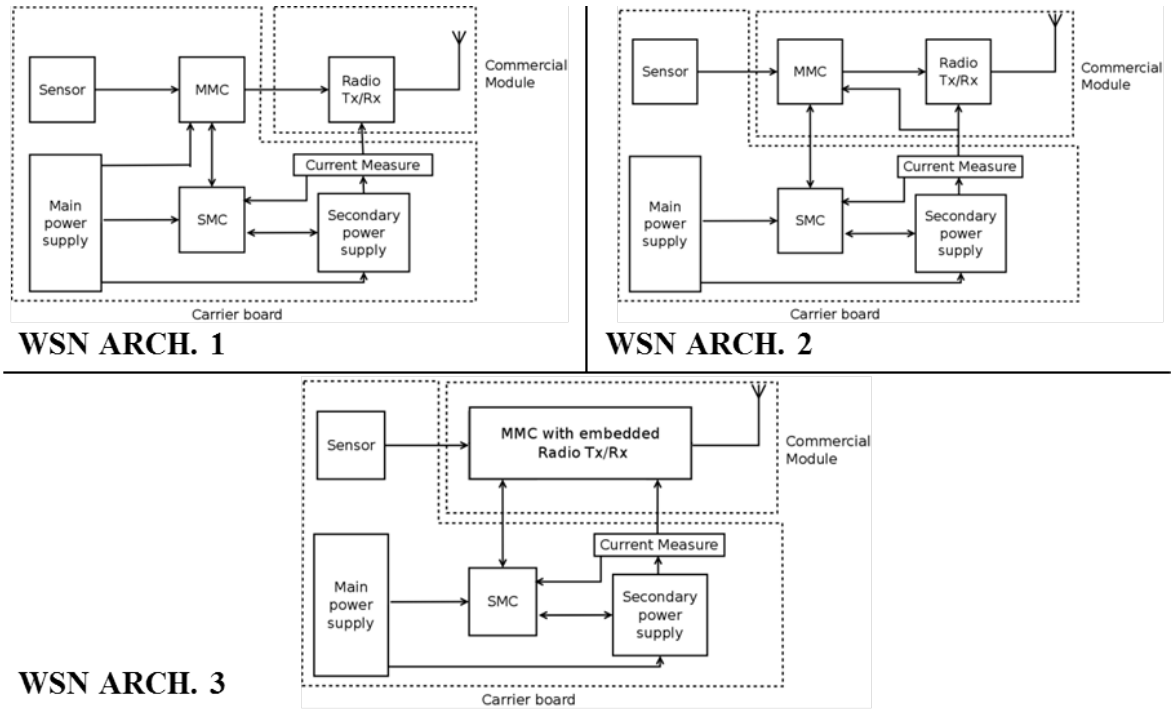


Figure 6.3: Possible architectures for modules safety ready using commercial WSN nodes

systematic errors by functional safety standards. Software quality must therefore consider the whole software life-cycle, because an error in software can be not only a classic bug e.g. a buffer overflow, but also a wrong or incomplete implementation of the software safety requirements. The purpose of this section is to review some open source embedded operating systems with regard to these quality criteria.

6.2.1 Software quality

ISO 25119 defines a Software Required Level (SRL) which is used to classify a given software component depending on its required reliability and ultimately, its safety. For each SRL ISO 25119 defines a set of measures to adopt that can be divided in three main categories:

- software safety requirements specification. The safety requirements should be derived from the technical safety concept (see ISO25119-3 §7.2). Requirements must be clearly defined, including response times of safety functions, and the manner in which they must be defined depends on the SRL, from informal methods (natural language) to formal methods (mathematical models and notation);

- software design and implementation. Here the goal is to write a readable, testable and maintainable source code and to specify in detail the behaviour of software. This is the most important part, and the requirements at the different SRL include programming language, algorithms and tools. These rules become more restrictive with higher SRL;

- software testing and validation. The objective is to verify that software safety requirements are fulfilled by the specific implementation. This part also provides a proof that requirements at machine level are qualified, complete and completely achieved;

6.3 OS Comparison

This section analyzes two widespread operating systems for WSN, ContikiOS and TinyOS, as well as a general-purpose hard real-time operating system, ChibiOS, with reference to the software quality criteria defined in section 6.2.1.

6.3.1 Existing reviews

A comparative overview of operating systems for WSN is given in [99]. Here the authors consider the real-time capabilities of the OS, among other things, but still are focused on typical WSN requirements, like for example power consumption, scheduling, communication protocols. Another recent comparison of WSN Operating Systems can be found in [100]. All these works, while providing results regarding the protocols to be used in safety-relevant wireless applications, or OS used in WSN applications, do not consider the impact of safety requirements on software implementation.

6.3.2 State of the art

The current market for WSN specific OSs offers several products with heterogeneous characteristics. One of the first OS published is TinyOS [101], that is strongly oriented to WSN nodes, in particular to power consumption. Another important OS is ContikiOS [102] which has been one of the first WSN OSs that provided IP communication by using the uIP TCP/IP stack. Other available OSs, just to name a few of them, are LiteOS [103], MagnetOS [104], AmbientRT[105], MANTIS OS [106] and SOS [107], some of which are not actively developed anymore. The main implementation characteristics that can be analysed are:

- **monolithic system vs modular system** Contiki is a modular system, where the different components are loosely coupled, while TinyOS is a monolithic system, where the different components are tightly coupled. A monolithic system will have a smaller footprint, that is an advantageous characteristic for WSN nodes that have limited memory sizes. On the other hand a modular system allows a better division between kernel and user space in order to grant a greater safety of the OS execution. The division here is considered only from a source code organisation point of view, since many hardware platforms for WSN don't support hardware separation mechanisms like MMU/MPU

or privilege levels. Additionally a modular system will allow both complete or partial modifications of the application through Over The Air Updates.

- **asynchronous programming paradigm vs synchronous**. Almost every WSN operating system is designed to be used through an event based programming paradigm, because it result in a very small implementation overhead, a highly responsive system and a power-saving solution. However event based programming can't be considered a safety compliant method because it does not give any real-time guarantees, because it is difficult to determine with certainty the worst case latency before the execution of the response to a safety critical event. In event based programming this is very difficult or not possible at all due to cooperative scheduling of the events. On the other hand a multi-threading system, usually implemented with a preemptive scheduler, is more suited for real-time applications because of higher control of timeouts of scheduled events and activities. Unfortunately, preemptive scheduling is not natively supported by most OSs (but is some cases can be supported through additional libraries, as, for example, TinyThread [108] or TOSThreads [109]).

- **static vs dynamic system** The ISO 25119 standard discourages the use of dynamic memory allocation because, particularly in an embedded system, in which the resources are very limited, the programmer can't be sure that the memory location required will be available in the moment in which it will be requested. So the error condition must be properly handled, but this could in turn result in an unacceptable performance degradation. On the other hand, a static system eliminates the possibility of such run-time errors, even if it can be a problem due to the small amount of memory that characterises WSN nodes. Considering the OSs named before, TinyOS is a static system, where the allocation of memory has to be performed at design time, while Contiki is a dynamic system.

The points analysed are the most important in order to analyse the current state of the art of WSN Operative Systems, with regard to safety critical applications. Many other aspects can be considered, and in the following paragraphs more technical details will be analysed in a more precise manner.

6.3.3 OS Safety requirements

In order to be used in a safety-related system, the software must be certified according to the relevant standards in the chosen application field. For example, considering the ISO 25119 standard, all the software that is needed to perform a safety-related function must be certified with a Software Required Level (SRL) from 0 (looser requirements) to 3 (stricter requirements),

depending on the Functional Safety Performance Level required for the application. The standard considers the use of an operating system as a component to create different execution domains, which can be used to separate the different tasks, especially the safety-relevant from the non-safety-relevant. This standard also mandates that the operating system, if used, must withstand a SRL equal or higher than the highest SRL of all the tasks handled by the OS. A common practice in safety-critical systems is using a pre-certified operating system, because the amount of work required to certify an operating system for high SRLs can be overwhelming. However, for lower SRL, it may be feasible to bear the certification costs together with the application software, provided that the OS already satisfies all or part of the SRL requirements. This approach can lead to lower costs for high production volumes.

The requirements that need to be satisfied are usually similar to high-reliability systems requirements, although it must be noted that reliability and safety are two different things. While reliability is related to the percentage of time the system operates as expected, safety is related to the residual error probability in a system, i.e. errors are admitted but they must be detected, and the system must be able to react accordingly, in order to avoid damages or injuries to persons, environments or other entities. The ISO 25119 has a series of requirements for how software must be developed, for every phase of the V-model, spanning from specification to implementation and testing. These requirements vary depending on the required SRL. In order to assess the overall software quality, a number of metrics are considered, described in the following sections. These metrics are then evaluated for a number of open-source operating systems, based on the information freely available, but without analysing the code base in depth. Also, here the safety requirement specifications are not considered, since for the considered OS there are no proper safety specifications, but in the best case there are functional tests. The specifications in this case can be seen as present proportionally to the implemented tests, but from the safety certification point of view, both for OS and for the entire application, the unavailability of OS Functional Safety Specifications is already a non-conformity issue.

Implementation

The software for embedded systems is usually implemented in the C language, or some language subset like MISRA-C 2012, which is sometimes considered a stronger typed subset of C. Many static analysers support MISRA C checking, so they can automatically find violations in the source code for many MISRA rules. This kind of analysis is here used as a quantitative measure to assess implementation quality; although a full MISRA check require considerably more effort than just running a static code analyser, like a manual inspection and rigorous rule deviations, for the purpose of this analysis this measure is considered sufficient and indicative of code quality.

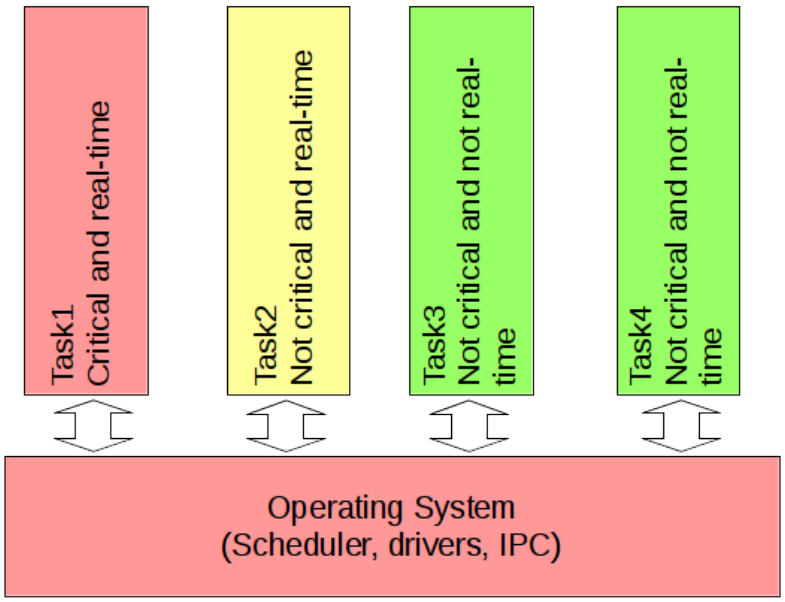


Figure 6.4: Domain separation for 4 tasks with different levels of criticality and real-time requirements. Colour indicated the level of criticality: red high, yellow medium, green none

Test Coverage

Test-driven development has become widespread both in proprietary and open source projects. In particular, the possibility to perform automated tests, for example after every single commit in the code base, is of great help to avoid regressions and ensure the correctness of the software functionality. It is common to define a percentage of test coverage, that is the percentage of software that is covered by the tests. There are many ways in which this percentage can be measured, the most common being the number of lines of code executed. Other metrics can be:

- the number of modules;
- the number of functionality defined in the specifications;
- the number of branches;

Domain Separation

An important functionality, desirable especially for mixed-criticality systems, is the ability to execute different tasks in different domains, where the operating systems guarantees the independence of these domains. An example of partitioning is described in appendix B of ISO 25119, where time, memory and communication resources are considered. The different domains can be executed in different regions of memory, by means of an hardware MPU

(Memory Protection Unit) or MMU (Memory Management Unit). The memory separation should include both code and data memory and also I/O memory, e.g. memory-mapped peripherals. If different tasks in different memory partitions need to share the same hardware peripheral, a proxy task must be used, forcing an appropriate policy for accessing the resource. Another important aspect is temporal independence of tasks; for instance, a low-priority task which takes a long time to execute must not prevent the correct execution of a fast task with higher priority. This is usually achieved using a priority-based preemptive scheduler with an appropriate priority assignment, depending for example on the criticality of the task or the deadline for periodic tasks. Finally, if domain separation is implemented, other shared resources like a communication stack or interface must be handled in a way that a malfunctioning task can not affect other tasks. An example of domain separation is depicted in Figure 6.4.

Real-time Performance

In safety-related systems, it is necessary to have a timely execution of a recovery or emergency procedure, which can not be delayed or prevented to execute. This guarantee can be given for example by a hard real-time operating system, often referred to as RTOS (Real-Time Operating System), with priority-based preemptive scheduling. Common priority schemes include the Rate Monotonic (RM) and Earliest Deadline First (EDF) [110]. It must be noted that also different concurrency models, if appropriately used, can result in predictable execution of critical tasks. For example, often the firmware of Electronic Control Units (ECU) has no operating system, but all the tasks are implemented in the main function and executed periodically. This is similar to having a cooperative scheduler for all the tasks. In this case, if the execution time of all the tasks has an upper bound, then the worst-case reaction time is also upper bounded. While this architecture is often used for safety-critical ECUs, it is difficult to use for WSNs, since it is not possible to use low-power modes. Additionally, it can become very difficult to perform a global analysis of the tasks in case of cooperative scheduling.

6.3.4 Evaluation of existing OS

Contiki-OS

Implementation ContikiOS is an operating system developed mainly in standard C and portable to various platforms. Its coding style is described in the documentation, and it covers only the formatting style. A MISRA C check has been performed, using the PC-Lint static analyser, with the default configuration for MISRA checking, enabling all MISRA rules and directives. A total of 1138 files has been checked, covering the *dev*, *core*, *platform* and *cpu* directories of the Contiki 3.0 source tree. The *apps* directory has not been analysed, as it

contains application-level modules. In total, 73.681 messages have been generated, most of them related to either styling issues or easily-correctable errors. However, there are many reports of more severe errors:

- Uninitialised variables (28 messages)
- Unused return codes from functions (284 messages)
- Null pointer dereferencing (13 messages)
- Multiple side effects in expressions (34 messages)
- Buffer overrun (12 messages)
- Shadowing of variables (44 messages)
- Discarding of *const* or *volatile* qualifiers (4 messages)
- Recursive functions (1 message)

Some messages were related to false positives given by the static analyser, while other has been manually verified, although not all results have been checked. Finally, a lot of messages were related either to violations of the MISRA data type model, much stricter about mixing variables of different types, or warnings about incompatible pointers and pointer arithmetic.

Test Coverage Regression tests are performed on every new version of ContikiOS, although they cover mostly the communication protocols and not the low-level operating system services like scheduling and inter-task communication.

Domain Separation Memory separation is not supported because of lack of support in most supported architectures (e.g. msp430, ARM Cortex M0). Its scheduler support both cooperative and preemptive tasks, but there are no well-defined mechanisms for inter-task communication; each module is free to expose its API, and concurrency issues must be handled manually.

Real-time Performance The scheduler does not support different priorities for the tasks, which can however be cooperative or preemptive. A number of timer modules are available, notably the rtimer module provides real-time timers, which is the only way to schedule preemptive tasks.

TinyOS

Implementation The TinyOS operating system is implemented using NesC, a language developed ad-hoc for WSN firmware. It supports asynchronous programming end energy saving, as well as a component-based organisation of the source code. The NesC compiler translates the source file in a C source file, which is then compiled with a normal C compiler. The NesC compiler also executes whole-program tests in order to check the concurrent access to resources. An extension of the C language has been developed for SafeTinyOS [111], where an additional step has been added to the compilation. In this case, NesC code is translated into annotated C code, which is then translated into classic C code. The annotated C code, to be used with the Deputy compiler, part of the Ivy Toolchain ([112] and [113]), has been developed with type safety in mind, and it allows to check pointers, arrays, unions, structures, string/array termination, as well dynamic features like memory allocation. Furthermore, the Heapsafe and SharC components of Ivy allow to execute whole-program checks for memory allocation and sharing between concurrent tasks. Although this certainly contribute to enhance the reliability of the firmware, it is a substantially different approach than MISRA C. The Ivy Toolchain appears more focused on the most common causes of implementation errors, that are pointers and type casts for C software, as well as sharing of resources for concurrent tasks. On the other hand MISRA C have a more general target, that is to remove the inconsistencies and weaknesses of the whole language. For example, the Deputy documentation does not consider undefined behaviours that can be present in the auto generated code, while the MISRA C guidelines cover all the aspects of the C language.

Test Coverage Unit testing is not part of the main TinyOS repository, but can be found on the *tinyos-2.x-contrib* repository [114], with a framework called TUnit. However it seems that tests for the operating system are not comprehensive, but the TUnit module is more focused on supporting the test of the final application.

Domain Separation Memory domains are not supported, because of lack of support in the main reference architectures (AVR and MSP430). Although TinyOS has been ported to architectures with Memory Protection Unit like ARM Cortex M, currently the MPU is not used. The separation of tasks in time is statically checked by the NesC compiler for race conditions, and these checks are improved using the Ivy Toolchain. Inter-task communication is handled asynchronously by means of events.

Real-time Performance TinyOS execution model is composed of events, which must be designed to run to completion. Although initially events were scheduled with a FIFO policy,

subsequent works added both priority-based and preemptive scheduling.

ChibiOS/RT

The ChibiOS project is composed of three major software components: Chibios/Nil, ChibiOS/RT are two RTOS, the first aimed at extremely low resource usage, but with limited features, while the second is slightly more resource-demanding, but with a more complete feature set. Finally, the ChibiOS/HAL is a set of peripheral drivers for various embedded platforms, which can be used theoretically with any RTOS, like ChibiOS/Nil or ChibiOS/RT, or even without RTOS. In the FACTOTHUMS project an experimental firmware using the commercial Zolertia Z1 module has been developed. Instead of using an existing WSN stack, we developed our own, using ChibiOS/RT as scheduler and operating system. The choice of a real-time operating system was done because the application needed precise timing both for the sensor reading and the WSN stack.

Implementation The ChibiOS/RT operating system is implemented in MISRA C 2012, with some justified exceptions, based on design decisions. A static code analysis with the PC-Lint analyser, on the ChibiOS 16.1.4 release, reported no violation of the checked MISRA rules, using the configuration provided with the source code. Additionally, the coding standard in terms of naming conventions and design patterns are explicit and consistently used on all the software. The documentation for the source code is available and generated with Doxygen, and includes a description of all the modules and their relationship.

Test Coverage The ChibiOS/RT kernel has a test suite to verify the functional correctness of mechanisms like priority-based scheduling, priority inversion, timers and all the synchronisation primitives offered by the operating system. The test suite can be executed both on a simulator and on real hardware, and can be used also as a benchmark of a given platform. The test suite is available for both the Nil and RT operating systems, and also for the HAL, so the peripheral drivers can also be tested on different hardware platforms.

Domain Separation The separation of the different threads is done in the time domain, that is with a priority-based preemptive scheduling. Even in this case, care must be taken using critical regions, which can introduce unexpected latency. The different tasks must then be developed in a coherent way, to avoid timing interference between different threads. The only execution model available is the single-process/multi-thread execution model, that is all tasks share the same addressing space, and memory separation is not implemented, even if available by means of a MPU or MMU peripheral on some of the supported architectures like ARM Cortex M3, M4, M7 and PowerPC e200x. Finally, the communication between different

tasks can be done with mailboxes with an asynchronous interface. This allows the decoupling of communication between different tasks.

Real-time Performance The scheduler of ChibiOS/RT is a priority-based preemptive scheduler, suited for hard-real time systems. The time overhead of operations like context-switch, interrupts and synchronisation primitives can be obtained using the test suite. There is support for performance statistics, so although not fully integrated it is possible to develop a task supervisor on top of it, to check for timing errors. However such supervisor is currently not implemented.

6.3.5 Discussion

The analysis reported here reflects the main development perspective of different operating systems, which in turn reflects the typical application requirements of different fields. Systems commonly used in IoT applications do not have the characteristics expected for safety-relevant applications, but are optimised for low-power and are tightly coupled with common WSN protocol stacks. This is in part expected, since wireless links are not usually used for safety applications, but on the other hand it means that all the software already existing will be in general difficult to adapt to the new requirements of safety applications. However an interesting result emerges, both for its practical and research impact: the substantially different approach to reliability for C programs. While automotive, aerospace and other fields have elaborated complete subsets of the C language, with formal verification and deviation rules in mind, addressing *all* aspects that leave room for errors, the Safe NesC approach is to introduce new language constructs on a case-by-case base, to handle error-prone operations by means of an additional level of translation of source code. This approach leads to non backwards-compatible source code, but at the same time leverages existing C compilers and put less burden on the programmer. Finally, as hard real-time systems are often associated to safety, an RTOS has more chance to satisfy the typical software safety requirements. In the case of ChibiOS, much work is publicly accessible, like test cases and MISRA C analysis configurations, as well as documentation. This can change for other open source RTOS, but generally are often backed by a company which can provide the necessary support for safety-related development (e.g. ChibiOS, Eriks Enterprise, FreeRTOS).

6.4 Experimental prototype

In section 6.1.4 three different kinds of architectures have been described in order to realise a Wireless Sensor Node for safety-related applications. The architectures are designed considering

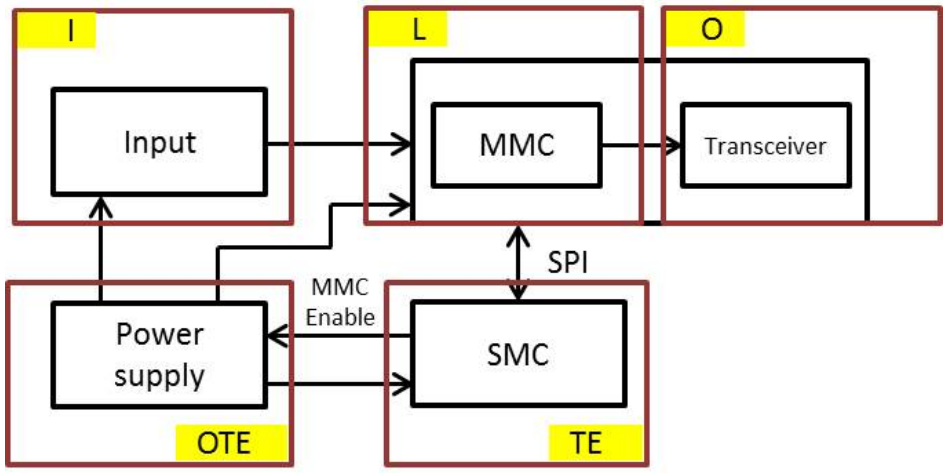


Figure 6.5: Architecture of SWSN module with reference to a hardware category 2 as described in ISO 25119.

the availability of commercial modules existing: transceiver-only module, transceiver module with on-chip microcontroller, microcontroller with integrated transceiver. The experimental SWSN node belongs to the third category. Figure 6.5 shows the block diagram of the SWSN architecture, which is easily attributable to an ISO25119 hardware category 2 considering the various building blocks. In Figure 6.6 the architecture is showed with greater detail; the module is battery powered and it uses three Low-Drop-Out regulators (LDO) to regulate the voltage that supplies the MMC/transceiver module, the SMC and input's pull-up network. The MMC/transceiver module is a commercial Embit EMB-Z2538 module, which in turn is composed of a CC2538 system-on-chip, based on ARM Cortex M3 architecture, with a IEEE 802.15.4 transceiver and an RF front-end. As a consequence the module represents both the logic (MMC) and the output.

The SMC block is composed of a MSP430F5324 SoC, connected to the MMC through a SPI line. This allows a continuous check of the status of the MMC logic. The checks will be software dependent, and include a smart watchdog, a task monitor, ALU tests, memory tests and so on, depending on the diagnostic coverage requested by the application. A second SMC to MMC diagnostic measure has been adopted: a current measure block, using a shunt resistor placed between the EMB-Z2538's LDO and the battery, in order to monitor the current absorbed. In this way it is possible to diagnose a possible transceiver fault, detecting a failure causing the abnormal transmission of packets. In fact, when the transceiver is transmitting or receiving, the current absorbed by the module is significantly increased. This allows the SMC to distinguish between the idle state (no transmission nor reception) and the working state of the module. If a working or idle state is detected for too long compared to the network duty-cycle of a normal operation, the module is considered in error and is turned off by the

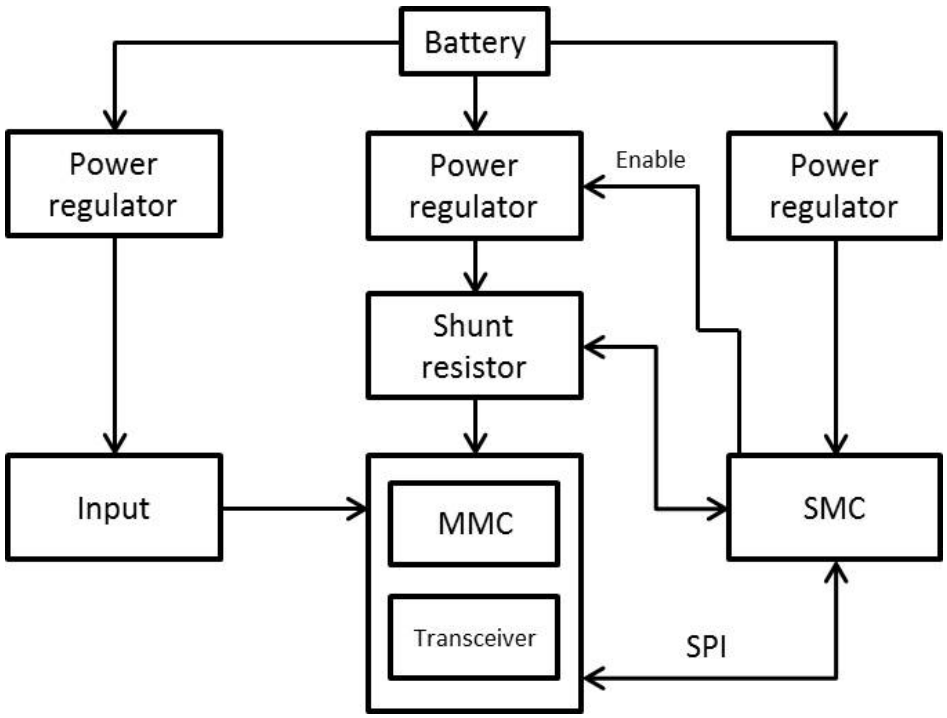


Figure 6.6: Detailed architecture of SWSN module.

SMC. It is worth noting that simply resetting the MMC in this case would not necessarily ensure that the transceiver is deactivated, as the very same transceiver deactivation logic from the reset path can be in fault.
As required by an hardware category 2 the SMC emergency action is to drive the system in a safe state in case an error is detected. In this architecture the safe state is reached by switching off the MMC/transceiver module, by means of EMB-Z2538 LDO's enable pin.

6.4.1 Power supply

As stated in the previous paragraph the node is battery powered but it has three power regulators. The choice has been to use three LDOs: one to supply the inputs' pull-up network, the second one to supply the wireless module and the last one to supply the SMC. The reasons to adopt three regulators are the following:

- The WSN module must have a dedicated power regulator in order to allow the SMC to switch it off through the enable pin in case any problem on the logic is detected. Moreover the chosen LDO support a maximum current of 200 mA and the peak current absorbed by the WSN module in transmission is 160 mA. So, connecting other devices to the same power regulator would put it near its limits, increasing the stress of the component and as a consequence the probability of failure.

- The inputs do not have a shared regulator with SMC because they are external components, so they can be wrongly connected or short-circuit can happen on the cable

connecting the sensors, causing the power regulator to switch off due to its internal protection. In all these situations the SMC can't be switched off, but must ensure that the safe state is maintained.

LDO regulators have been chosen instead of switching regulators for two reasons:

- due to the low voltage provided by the couple of AAA batteries, it is not tolerable to have a big drop-out voltage;
- it is not possible to use switching power regulators because of their noisy nature.

In fact, the power supply should be as clean as possible, as stated in EMB-Z2538 module's datasheet. It has also been excluded to connect the components directly to the batteries because in this case a mechanism to turn off the MMC/radio block would miss. Regarding the batteries it has been chosen to use two AAA ones. In this kind of applications the most relevant factor for the choice of batteries is the size, considering that WSN modules has to be as small as possible. This module does not use Li-ion coin batteries because they can supply a very small peak current, while the transceiver can require up to 160 mA peak current during transmission. The choice to use only two AAA batteries and not a higher number is due to space reasons, but caused the power supply to be at a low voltage. In fact we have a 3V battery package regulated by LDOs to supply 2.5 V to the components.

6.4.2 Main microcontroller

The MMC role is to acquire the inputs and to send data periodically through wireless sensor network. This is the main component of the board and it consist of an Embit EMB-Z2538 module. The reasons that led to the choice of this module are mainly three:

- it has an integrated RF front end;
- it is based on an ARM Cortex M3 architecture which is powerful enough to handle even complex operations;
- it has a crystal oscillator, which is fundamental to have the precise low-jitter timing required for low-latency networks.

In fact, previous experience at CNR-IMAMOTER with COTS boards (Zolertia Z1) using an RC oscillator showed that the oscillator jitter limited the achievable real-time performance, and shifts of up to $200\mu s$ on a time slot were measured. The integrated RF front end is very important in order to reduce design effort given by the typical complexity of radio-frequency circuits. Using a widely adopted architecture such as ARM Cortex M3 grants a good level of support with compilers, debuggers and development tools in general. For example, in addition

to the standard C toolchain available from many vendors, even the GNAT Ada toolchain is available for ARM Cortex M, which GPL version can be freely downloaded from either Adacore or compiled from upstream GCC, and which includes support for the Ravenscar profile, offering services usually found in C RTOS. This means that even tools for high-integrity systems are already available and may be used in future. The only drawback of using an integrated module like EMB-Z2538 is the limited availability of input pins; in fact only a small number of analog and digital I/O pins are available. To overcome this limitation and be able to connect a higher number of sensors to the module, it has been necessary to add analog multiplexers with low series resistance as an additional component.

6.4.3 Safety microcontroller

As SMC a Texas instruments MSP430F5324 has been chosen. This microcontroller family has one of the lowest power consumption available on the market, and is widely used on COTS WSN nodes. Furthermore, it has a 12 bits ADC, necessary to acquire with sufficient precision the voltage on the shunt resistor and measure the current consumption of the MMC/radio block.

Chapter 7

Conclusions

A complete software, hardware and protocol co-design for Wireless Sensor Network modules has been proposed in this thesis, representing a reference upon which safe WSN can be built. One of the major contribution regarding hardware architectures is identifying the output of the system as the physical packet transmission, which allows to define the safe state as a de-energization of the radio transceiver, blocking any further activity and decoupling it from the effective content of the transmitted packet. This allows a standard hardware category 2 to be used, according to ISO 25119-3, and potentially already reaching a much greater protection against undetected errors than COTS hardware modules. However, with the aim of allowing a functional safety analysis to be performed, the hardware category alone is not sufficient, but must correspond to a certain Software Reliability Level and a certain Diagnostic Coverage. For the first point, it has been assessed that existing WSN operating systems must undergo a relevant modification with regard to documentation and in general software development life-cycle. It may even be convenient to start from a completely different operating system, one designed for hard real-time applications, and port the protocol stack to it, since in this case the operating system would probably already have many desired characteristics such as proper documentation, module and integration test, explicit coding styles and MISRA C. For the second point, it was noted that real-time communications on a WSN are already possible with the operating modes of IEEE 802.15.4e, like LLDN, DSME and TSCH; however this thesis proposed an enhanced version of LLDN which allows to reduce the worst-case latency or to increase the total number of sensors used in a single network, providing an analytic analysis and comparison with standard LLDN. Finally, the configuration time of the WSN is analysed by means of simulations, with the purpose of assessing the feasibility of a multi-configuration WSN, which can be very useful if different sub-regions of sub-groups of nodes can be identified, allowing a safe WSN to scale with the number of nodes.

As a result, a prototype implementation of a safe WSN node has been realised, applying the

concepts for hardware, software and protocols co-design.

Bibliography

[1] ISO. *ISO 61508:2010: Functional safety of electrical/electronic/ programmable electronic safety-related systems*. Geneva, Switzerland: International Organization for Standardization, 2010.

[2] "Functional Safety: A Straightforward Guide to Applying IEC 61508 and Related Standards". In: *Functional Safety (Second Edition)*. Ed. by David J Smith and Kenneth G L Simpson. Second Edition. Oxford: Butterworth-Heinemann, 2004.

[3] ISO. *ISO 12100:2010: Safety of machinery – General principles for design – Risk assessment and risk reduction*. Geneva, Switzerland: International Organization for Standardization, 2010.

[4] ISO. *ISO 25119:2010: Tractors and machinery for agriculture and forestry — Safety-related parts of control systems (all parts)*. Geneva, Switzerland: International Organization for Standardization, 2010.

[5] ISO. *ISO 15998:2008: Earth-moving machinery – Machine-control systems (MCS) using electronic components – Performance criteria and tests for functional safety*. Geneva, Switzerland: International Organization for Standardization, 2008.

[6] Motor Industry Software Reliability Association and Motor Industry Software Reliability Association Staff. *MISRA C:2012: Guidelines for the Use of the C Language in Critical Systems*. Motor Industry Research Association, 2013. ISBN: 9781906400101.

[7] ISO. *ISO 11898:2015: Road vehicles – Controller area network (CAN)*. Geneva, Switzerland: International Organization for Standardization, 2015.

[8] ISO. *ISO 11783:2014: Tractors and machinery for agriculture and forestry – Serial control and communications data network (All Parts)*. 2014.

[9] Chris Valasek and Charlie Miller. "Adventures in Automotive Networks and Control Units". In: *DEFCON 23*. Las Vegas, 2015.

[10] Chris Valasek and Charlie Miller. *After Jeep Hack, Chrysler Recalls 1.4M Vehicles for Bug Fix*. Online. July 2015. URL: https://www.wired.com/2015/07/jeep-hack-chrysler-recalls-1-4m-vehicles-bug-fix/.

[11] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. "Experimental Security Analysis of a Modern Automobile". In: *Proceedings of the 2010 IEEE Symposium on Security and Privacy*. SP '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 447–462. ISBN: 978-0-7695-4035-1. DOI: 10.1109/SP.2010.34. URL: http://dx.doi.org/10.1109/SP.2010.34.

[12] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. "Comprehensive Experimental Analyses of Automotive Attack Surfaces". In: *Proceedings of the 20th USENIX Conference on Security*. SEC'11. San Francisco, CA: USENIX Association, 2011, pp. 6–6. URL: http://dl.acm.org/citation.cfm?id=2028067.2028073.

[13] Chris Valasek and Charlie Miller. *A Survey of Remote Automotive Attack Surfaces*. Tech. rep. IOActive Technical White Paper, 2014.

[14] Chris Valasek and Charlie Miller. *Hackers Remotely Kill a Jeep on the Highway. With Me in It*. Online. July 2015. URL: http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/.

[15] Mohammad Al-Shurman, Seong-Moo Yoo, and Seungjin Park. "Black Hole Attack in Mobile Ad Hoc Networks". In: *Proceedings of the 42Nd Annual Southeast Regional Conference*. ACM-SE 42. Huntsville, Alabama: ACM, 2004, pp. 96–97. ISBN: 1-58113-870-9. DOI: 10.1145/986537.986560. URL: http://doi.acm.org/10.1145/986537.986560.

[16] ISO. *ISO 15765:2011: Road vehicles — Diagnostic communication over Controller Area Network (DoCAN)*. 2011.

[17] SAE. *SAE J1939: Recommended Practice for a Serial Control and Communications Vehicle Network*. Warrendale, PA: Society of Automotive Engineers, 2013.

[18] ISO. *ISO 11783:2014: Tractors and machinery for agriculture and forestry – Serial control and communications data network - Part 3: Data Link Layer*. Tech. rep. 2014.

[19] H. Kleinknecht. *CAN Calibration Protocol Version 2.1, Technical report*. Tech. rep. Standardization of Application/Calibration Systems task force, 1999.

[20] ISO. *ISO 14230:2013: Road Vehicles - Diagnostic systems - Keyword Protocol 2000 - Part 3: Application Layer*. 2013.

[21] Anthony Van Herrewege, Dave Singelee, and Ingrid Verbauwhede. "CANAuth - A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus". In: *ECRYPT Workshop on Lightweight Cryptography*. Nov. 2011, pp. 229–235.

[22] Morris J. Dworkin. *SP 800-38B. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*. Tech. rep. Gaithersburg, MD, United States: National Institute of Standards and Technology, 2005.

[23] C. Szilagyi and P. Koopman. "Flexible multicast authentication for time-triggered embedded control network applications". In: *2009 IEEE/IFIP International Conference on Dependable Systems Networks*. June 2009, pp. 165–174. DOI: 10.1109/DSN.2009.5270342.

[24] Andreea-Ina Radu and Flavio D. Garcia. "LeiA: A Lightweight Authentication Protocol for CAN." In: *21st European Symposium on Research in Computer Security (ESORICS 2016)*. Heraklion, Crete, Greece, 2016.

[25] Paula Vasile, Bogdan Groza, and Stefan Murvay. "Performance Analysis of Broadcast Authentication Protocols on CAN-FD and FlexRay". In: *Proceedings of the WESS'15: Workshop on Embedded Systems Security*. WESS'15. Amsterdam, Netherlands: ACM, 2015, 7:1–7:8. ISBN: 978-1-4503-3667-3. DOI: 10.1145/2818362.2818369. URL: http://doi.acm.org/10.1145/2818362.2818369.

[26] Ryo Kurachi, Yutaka Matsubara, Hiroaki takada, Naoki Adachi, Yukihiro Miyashita, and Satoshi Horihata. "CaCAN - Centralized Authentication System in CAN (Controller Area Network)". In: *Embedded Security in Cars (ESCAR)*. 2014.

[27] C. W. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli. "Security-Aware Modeling and Efficient Mapping for CAN-Based Real-Time Distributed Automotive Systems". In: *IEEE Embedded Systems Letters* 7.1 (Mar. 2015), pp. 11–14. ISSN: 1943-0663. DOI: 10.1109/LES.2014.2354011.

[28] ECRYPT. *Yearly Report on Algorithms and Keysizes*. Tech. rep. ECRYPT II, 2012.

[29] "IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks". In: *IEEE Std 802.1Q-2014 (Revision of IEEE Std 802.1Q-2011)* (Dec. 2014), pp. 1–1832. DOI: 10.1109/IEEESTD.2014.6991462.

[30] Michael Düll, Björn Haase, Gesine Hinterwälder, Michael Hutter, Christof Paar, Ana Helena Sánchez, and Peter Schwabe. "High-speed Curve25519 on 8-bit, 16-bit, and 32-bit Microcontrollers". In: *Des. Codes Cryptography* 77.2-3 (Dec. 2015), pp. 493–514. ISSN: 0925-1022. DOI: 10.1007/s10623-015-0087-1. URL: http://dx.doi.org/10.1007/s10623-015-0087-1.

[31] H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104 (Informational). Updated by RFC 6151. Internet Engineering Task Force, Feb. 1997. URL: http://www.ietf.org/rfc/rfc2104.txt.

[32] J. H. Saltzer, D. P. Reed, and D. D. Clark. “End-to-end Arguments in System Design”. In: *ACM Trans. Comput. Syst.* 2.4 (Nov. 1984), pp. 277–288. ISSN: 0734-2071. DOI: 10.1145/357401.357402. URL: http://doi.acm.org/10.1145/357401.357402.

[33] Yee Wei Law, Jeroen Doumen, and Pieter Hartel. “Survey and Benchmark of Block Ciphers for Wireless Sensor Networks”. In: *ACM Trans. Sen. Netw.* 2.1 (Feb. 2006), pp. 65–93. ISSN: 1550-4859. DOI: 10.1145/1138127.1138130. URL: http://doi.acm.org/10.1145/1138127.1138130.

[34] C. Mickael and M Kevin M.and Marine. “Survey and Benchmark of Lightweight Block Ciphers for Wireless Sensor Networks”. In: *IACR Cryptology aPrint Archive* (2013).

[35] D. S. Abdul Elminaam, H. M. Abdul Kader, and M. M. HadHoud. “Performance Evaluation of Symmetric Encryption Algorithms”. In: *Communications of the IBIMA* (2009).

[36] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. “A View of Cloud Computing”. In: *Commun. ACM* 53.4 (Apr. 2010), pp. 50–58. ISSN: 0001-0782. DOI: 10.1145/1721654.1721672. URL: http://doi.acm.org/10.1145/1721654.1721672.

[37] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff. “A Brief History of the Internet”. In: *SIGCOMM Comput. Commun. Rev.* 39.5 (Oct. 2009), pp. 22–31. ISSN: 0146-4833. DOI: 10.1145/1629607.1629613. URL: http://doi.acm.org/10.1145/1629607.1629613.

[38] Nicholas Weaver, Vern Paxson, Stuart Staniford, and Robert Cunningham. “A Taxonomy of Computer Worms”. In: *Proceedings of the 2003 ACM Workshop on Rapid Malcode*. WORM ’03. Washington, DC, USA: ACM, 2003, pp. 11–18. ISBN: 1-58113-785-0. DOI: 10.1145/948187.948190. URL: http://doi.acm.org/10.1145/948187.948190.

[39] Nick Feamster, Jennifer Rexford, and Ellen Zegura. “The Road to SDN: An Intellectual History of Programmable Networks”. In: *SIGCOMM Comput. Commun. Rev.* 44.2 (Apr. 2014), pp. 87–98. ISSN: 0146-4833. DOI: 10.1145/2602204.2602219. URL: http://doi.acm.org/10.1145/2602204.2602219.

[40] Kirill Mechitov and Gul Agha. “Software Service and Application Engineering”. In: ed. by Maritta Heisel. Berlin, Heidelberg: Springer-Verlag, 2012. Chap. An Architecture for Dynamic Service-oriented Computing in Networked Embedded Systems, pp. 147–164. ISBN: 978-3-642-30834-5. URL: http://dl.acm.org/citation.cfm?id=2363395.2363405.

[41] Alessio Botta, Walter de Donato, Valerio Persico, and Antonio Pescapé. "Integration of Cloud computing and Internet of Things: A survey". In: *Future Generation Computer Systems* 56 (2016), pp. 684–700. ISSN: 0167-739X. DOI: http://dx.doi.org/10.1016/j.future.2015.09.021. URL: http://www.sciencedirect.com/science/article/pii/S0167739X15003015.

[42] Thomas Erl. *SOA: Principles of Service Design*. Prentice Hall, 2007.

[43] Orna Agmon Ben-Yehuda, Muli Ben-Yehuda, Assaf Schuster, and Dan Tsafrir. "The Rise of RaaS: The Resource-as-a-service Cloud". In: *Commun. ACM* 57.7 (July 2014), pp. 76–84. ISSN: 0001-0782. DOI: 10.1145/2627422. URL: http://doi.acm.org/10.1145/2627422.

[44] M. Engelsberger and T. Greiner. "Application-independent approach for the dynamic management of IT-resources in cyber-physical systems". In: *2016 IEEE International Conference on Industrial Technology (ICIT)*. Mar. 2016, pp. 830–835. DOI: 10.1109/ICIT.2016.7474859.

[45] L. Wang, R. Wakikawa, R. Kuntz, R. Vuyyuru, and L. Zhang. "Data naming in Vehicle-to-Vehicle communications". In: *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*. Mar. 2012, pp. 328–333. DOI: 10.1109/INFCOMW.2012.6193515.

[46] Z. Yan, S. Zeadally, and Y. J. Park. "A Novel Vehicular Information Network Architecture Based on Named Data Networking (NDN)". In: *IEEE Internet of Things Journal* 1.6 (Dec. 2014), pp. 525–532. ISSN: 2327-4662. DOI: 10.1109/JIOT.2014.2354294.

[47] A. Bazzi, B. M. Masini, A. Zanella, C. De Castro, C. Raffaelli, and O. Andrisano. "Cellular aided vehicular named data networking". In: *2014 International Conference on Connected Vehicles and Expo (ICCVE)*. Nov. 2014, pp. 747–752. DOI: 10.1109/ICCVE.2014.7297650.

[48] W. Quan, C. Xu, J. Guan, H. Zhang, and L. A. Grieco. "Social cooperation for information-centric multimedia streaming in highway VANETs". In: *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*. June 2014, pp. 1–6. DOI: 10.1109/WoWMoM.2014.6918992.

[49] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. "A survey of information-centric networking". In: *IEEE Communications Magazine* 50.7 (July 2012), pp. 26–36. ISSN: 0163-6804. DOI: 10.1109/MCOM.2012.6231276.

[50] Roy Thomas Fielding. "Architectural Styles and the Design of Network-based Software Architectures". AAI9980887. PhD thesis. 2000. ISBN: 0-599-87118-0.

[51] W3C. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. W3C Recommendation 27 April 2007.

[52] *Introduction to AWS Security by Design*. November 2015. URL: https://d0.awsstatic.com/whitepapers/compliance/Intro_to_Security_by_Design.pdf.

[53] Martin Böhner, Alexander Mattausch, and Alexander Much. "Extending Software Architectures from Safety to Security". In: *Lecture Notes on Informatics (LNI), Gesellschaft für Informatik*. 2015, pp. 41–54.

[54] Ethernet POWERLINK Standardization Group (EPSG). *openSAFETY: The first open and bus-independent safety standard for all Industrial Ethernet solutions*. URL: http://www.open-safety.org.

[55] Josef Börcsök and Evzudin Ugljesa. "2Oo4 Architecture, an Advanced Processing Architecture for Safety Related Systems". In: *Proceedings of the 6th WSEAS International Conference on Applied Computer Science*. ACS'06. Canary Islands, Spain: World Scientific, Engineering Academy, and Society (WSEAS), 2006, pp. 119–125. ISBN: 960-8457-57-2. URL: http://dl.acm.org/citation.cfm?id=1973812.1973837.

[56] "IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer". In: *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)* (Apr. 2012), pp. 1–225. DOI: 10.1109/IEEESTD.2012.6185525.

[57] "IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)". In: *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)* (Sept. 2011), pp. 1–314. DOI: 10.1109/IEEESTD.2011.6012487.

[58] J. Drozdowicz and G. Mazurek. "IEEE 802.15.4 compliant in-building wireless sensor network". In: *Signal Processing Symposium (SPSympo), 2015*. June 2015, pp. 1–5. DOI: 10.1109/SPS.2015.7168267.

[59] J. Edwards. "Wireless Sensors Relay Medical Insight to Patients and Caregivers [Special Reports]". In: *IEEE Signal Processing Magazine* 29.3 (May 2012), pp. 8–12. ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2183489.

[60] M. M. Rodgers, V. M. Pai, and R. S. Conroy. "Recent Advances in Wearable Sensors for Health Monitoring". In: *IEEE Sensors Journal* 15.6 (June 2015), pp. 3119–3126. ISSN: 1530-437X. DOI: 10.1109/JSEN.2014.2357257.

[61] S. C. Mukhopadhyay. "Wearable Sensors for Human Activity Monitoring: A Review". In: *IEEE Sensors Journal* 15.3 (Mar. 2015), pp. 1321–1330. ISSN: 1530-437X. DOI: 10.1109/JSEN.2014.2370945.

[62] Weijun Tao, Tao Liu, Rencheng Zheng, and Hutian Feng. "Gait Analysis Using Wearable Sensors". In: *Sensors* 12.2 (Feb. 2012), pp. 2255–2283. ISSN: 1424-8220.

[63] C-C Yang and Y-L Hsu. "A Review of Accelerometry-Based Wearable Motion Detectors for Physical Activity Monitoring". In: *Sensors* 10.8 (Aug. 2010), pp. 7772–7788.

[64] D. B. Smith, D. Miniutti, T. A. Lamahewa, and L. W. Hanlen. "Propagation Models for Body-Area Networks: A Survey and New Outlook". In: *IEEE Antennas and Propagation Magazine* 55.5 (Oct. 2013), pp. 97–117. ISSN: 1045-9243. DOI: 10.1109/MAP.2013.6735479.

[65] K.Y. Yazdandoost and K. Sayrafian-Pou. *Channel model for body area network (BAN).* Tech. rep. IEEE p802.15-08-0780-09-0006. IEEE 802.15 Working Group Document, Apr. 2009.

[66] S. van Roy, F. Quitin, L. Liu, C. Oestges, F. Horlin, J. M. Dricot, and P. De Doncker. "Dynamic Channel Modeling for Multi-Sensor Body Area Networks". In: *IEEE Transactions on Antennas and Propagation* 61.4 (Apr. 2013), pp. 2200–2208. ISSN: 0018-926X. DOI: 10.1109/TAP.2012.2231917.

[67] Freescale. *KW0x SoC family.* [Online; accessed 27-September-2016]. URL: http://www.freescale.com/products/arm-processors/kinetis-cortex-m/w-series/kinetis-kw0x-48-mhz-sub-1-ghz-radio-ultra-low-power-wireless-microcontrollers-mcus:KW0x%5C#pspFeatures.

[68] Linear. *LTC5800-WHM SoC.* [Online; accessed 27-September-2016]. URL: http://www.linear.com/product/LTC5800-WHM.

[69] M. Petrova, J. Riihijarvi, P. Mahonen, and S. Labella. "Performance study of IEEE 802.15.4 using measurements and simulations". In: *IEEE Wireless Communications and Networking Conference, 2006. WCNC 2006.* Vol. 1. Apr. 2006, pp. 487–492. DOI: 10.1109/WCNC.2006.1683512.

[70] Decawave. *DWM1000 Module.* [Online; accessed 27-September-2016]. URL: http://www.decawave.com/products/dwm1000-module.

[71] Texas Instruments. *LTC5800-WHM SoC.* [Online; accessed 27-September-2016]. URL: http://www.linear.com/product/LTC5800-WHM.

[72] STE ksolutions. *micro.SP Technology.* [Online; accessed 27-September-2016]. URL: http://www.microsp.it/.

[73] Microchip. *RN2483, LoRa sub-GHz enabled SoC.* [Online; accessed 27-September-2016]. URL: http://www.microchip.com/wwwproducts/Devices.aspx?product=RN2483.

[74] Nordic Semicondictor. *nRF51422*. [Online; accessed 27-September-2016]. URL: https://www.nordicsemi.com/eng/Products/ANT/nRF51422.

[75] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. RFC 4944 (Proposed Standard). Updated by RFCs 6282, 6775. Internet Engineering Task Force, Sept. 2007. URL: http://www.ietf.org/rfc/rfc4944.txt.

[76] J. Hui and P. Thubert. *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*. RFC 6282 (Proposed Standard). Internet Engineering Task Force, Sept. 2011. URL: http://www.ietf.org/rfc/rfc6282.txt.

[77] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann. *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*. RFC 6775 (Proposed Standard). Internet Engineering Task Force, Nov. 2012. URL: http://www.ietf.org/rfc/rfc6775.txt.

[78] T. Watteyne, M. Palattella, and L. Grieco. *Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement*. RFC 7554 (Informational). Internet Engineering Task Force, May 2015. URL: http://www.ietf.org/rfc/rfc7554.txt.

[79] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, and R. Alexander. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. RFC 6550 (Proposed Standard). Internet Engineering Task Force, Mar. 2012. URL: http://www.ietf.org/rfc/rfc6550.txt.

[80] Aishwarya Parasuram. "An Analysis of the RPL Routing Standard for Low Power and Lossy Networks". MA thesis. EECS Department, University of California, Berkeley, May 2016. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-106.html.

[81] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252 (Proposed Standard). Updated by RFC 7959. Internet Engineering Task Force, June 2014. URL: http://www.ietf.org/rfc/rfc7252.txt.

[82] V. Honguntikar and G. S. Biradar. "Performance analysis of GTS allocation in IEEE 802.15.4 with CSMA/CA for Wireless Sensor Network". In: *2015 International Conference on Computers, Communications, and Systems (ICCCS)*. Nov. 2015, pp. 103–107. DOI: 10.1109/CCOMS.2015.7562881.

[83] G. Bhatti, A. Mehta, Z. Sahinoglu, J. Zhang, and R. Viswanathan. "Modified Beacon-Enabled IEEE 802.15.4 MAC for Lower Latency". In: *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*. Nov. 2008, pp. 1–5. DOI: 10.1109/GLOCOM.2008.ECP.173.

[84] Seungku Kim, Jae-Ho Lee, Hyosun Hwang, ChiSung Bae, and Doo-Seop Eom. "AB-MAC: A TDMA-based adaptive beaconing MAC protocol for body sensor networks". In: *Enabling Technologies for Smartphone and Internet of Things (ETSIoT), 2012 First IEEE Workshop on*. June 2012, pp. 1–6. DOI: 10.1109/ETSIoT.2012.6311257.

[85] G. Patti and L. Lo Bello. "A Priority-Aware Multichannel Adaptive Framework for the IEEE 802.15.4e-LLDN". In: *IEEE Transactions on Industrial Electronics* 63.10 (Oct. 2016), pp. 6360–6370. ISSN: 0278-0046. DOI: 10.1109/TIE.2016.2573754.

[86] G. Chalhoub, N. Hadid, A. Guitton, and M. Misson. "Deference Mechanisms Significantly Increase the MAC Delay of Slotted CSMA/CA". In: *Communications, 2009. ICC '09. IEEE International Conference on*. June 2009, pp. 1–5. DOI: 10.1109/ICC.2009.5198886.

[87] L. Kleinrock and F.A. Tobagi. "Packet Switching in Radio Channels: Part I–Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics". In: *Communications, IEEE Transactions on* 23.12 (Dec. 1975), pp. 1400–1416. ISSN: 0090-6778. DOI: 10.1109/TCOM.1975.1092768.

[88] Feng Wang, Dou Li, and Yuping Zhao. "Analysis and Compare of Slotted and Unslotted CSMA in IEEE 802.15.4". In: *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*. Sept. 2009, pp. 1–5. DOI: 10.1109/WICOM.2009.5303580.

[89] Zhan Jie and Liu Hongli. "Access Delay Analysis of ZigBee Protocol with Delay Line". In: *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*. Sept. 2007, pp. 6452–6455. DOI: 10.1109/WICOM.2007.1583.

[90] S. Pollin, M. Ergen, S. Ergen, B. Bougard, L. Der Perre, I. Moerman, A. Bahai, P. Varaiya, and F. Catthoor. "Performance Analysis of Slotted Carrier Sense IEEE 802.15.4 Medium Access Layer". In: *Wireless Communications, IEEE Transactions on* 7.9 (Sept. 2008), pp. 3359–3371. ISSN: 1536-1276. DOI: 10.1109/TWC.2008.060057.

[91] Y. S. Chen and Y. W. Lin. "C-MAC: An Energy-Efficient MAC Scheme Using Chinese-Remainder-Theorem for Wireless Sensor Networks". In: *2007 IEEE International Conference on Communications*. June 2007, pp. 3576–3581. DOI: 10.1109/ICC.2007.590.

[92] M. Baleani, A. Ferrari, L. Mangeruca, A. Sangiovanni-Vincentelli, Maurizio Peri, and Saverio Pezzini. "Fault-tolerant Platforms for Automotive Safety-critical Applications". In: *Proceedings of the 2003 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*. CASES '03. San Jose, California, USA: ACM, 2003, pp. 170–177. ISBN: 1-58113-676-5. DOI: 10.1145/951710.951734. URL: http://doi.acm.org/10.1145/951710.951734.

[93] A Ferrari, S Garue, M. Peri, S. Pezzini, L. Valsecchi, F. Andretta, and W. Nesci. "Design and implementation of a dual processor platform for power-train systems". In: *Proceedings of the Convergence 2000 International Congress on Transportation Electronics*. SAE, Oct. 2000.

[94] R. Isermann, R. Schwarz, and S. Stolzl. "Fault-tolerant drive-by-wire systems". In: *IEEE Control Systems* 22.5 (Oct. 2002), pp. 64–81. ISSN: 1066-033X. DOI: 10.1109/MCS.2002.1035218.

[95] Texas Instruments. *CC2538 SoC*. URL: http://www.ti.com/product/CC2538.

[96] Bluegiga. *BLE112*. URL: https://www.bluegiga.com/en-US/products/ble112-bluetooth-smart-module/.

[97] Zolertia. *Z1*. URL: http://zolertia.io/z1.

[98] Texas Instruments. *CC2520 transceiver*. URL: http://www.ti.com/product/CC2520.

[99] Muhammad Omer Farooq and Thomas Kunz. "Operating Systems for Wireless Sensor Networks: A Survey". In: *Sensors* 11.6 (2011), p. 5900. ISSN: 1424-8220. DOI: 10.3390/s110605900. URL: http://www.mdpi.com/1424-8220/11/6/5900.

[100] T. Vu Chien, H. Nguyen Chan, and T. Nguyen Huu. "A comparative study on operating system for Wireless Sensor Networks". In: *Advanced Computer Science and Information System (ICACSIS), 2011 International Conference on*. Dec. 2011, pp. 73–78.

[101] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. "System Architecture Directions for Networked Sensors". In: *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS IX. Cambridge, Massachusetts, USA: ACM, 2000, pp. 93–104. ISBN: 1-58113-317-0. DOI: 10.1145/378993.379006. URL: http://doi.acm.org/10.1145/378993.379006.

[102] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors". In: *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*. LCN '04. Washington,

DC, USA: IEEE Computer Society, 2004, pp. 455–462. ISBN: 0-7695-2260-2. DOI: 10.1109/LCN.2004.38. URL: http://dx.doi.org/10.1109/LCN.2004.38.

[103] Qing Cao, Tarek Abdelzaher, John Stankovic, and Tian He. "The LiteOS Operating System: Towards Unix-Like Abstractions for Wireless Sensor Networks". In: *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*. IPSN '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 233–244. ISBN: 978-0-7695-3157-1. DOI: 10.1109/IPSN.2008.54. URL: http://dx.doi.org/10.1109/IPSN.2008.54.

[104] Rimon Barr, John C. Bicket, Daniel S. Dantas, Bowei Du, T. W. Danny Kim, Bing Zhou, and Emin Gün Sirer. "On the Need for System-level Support for Ad Hoc and Sensor Networks". In: *SIGOPS Oper. Syst. Rev.* 36.2 (Apr. 2002), pp. 1–5. ISSN: 0163-5980. DOI: 10.1145/509526.509528. URL: http://doi.acm.org/10.1145/509526.509528.

[105] T. J. Hofmeijer, S. O. Dulman, P. G. Jansen, and P. J. M. Havinga. "AmbientRT - real time system software support for data centric sensor networks". In: *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004*. Dec. 2004, pp. 61–66. DOI: 10.1109/ISSNIP.2004.1417438.

[106] H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker, J. Deng, and R. Han. "MANTIS: System Support for multimodAl NeTworks of In-situ Sensors". In: *Proceedings of the 2Nd ACM International Conference on Wireless Sensor Networks and Applications*. WSNA '03. San Diego, CA, USA: ACM, 2003, pp. 50–59. ISBN: 1-58113-764-8. DOI: 10.1145/941350.941358. URL: http://doi.acm.org/10.1145/941350.941358.

[107] Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, and Mani Srivastava. "A Dynamic Operating System for Sensor Nodes". In: *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*. MobiSys '05. Seattle, Washington: ACM, 2005, pp. 163–176. ISBN: 1-931971-31-5. DOI: 10.1145/1067170.1067188. URL: http://doi.acm.org/10.1145/1067170.1067188.

[108] William P. McCartney and Nigamanth Sridhar. "Abstractions for Safe Concurrent Programming in Networked Embedded Systems". In: *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*. SenSys '06. Boulder, Colorado, USA: ACM, 2006, pp. 167–180. ISBN: 1-59593-343-3. DOI: 10.1145/1182807.1182825. URL: http://doi.acm.org/10.1145/1182807.1182825.

[109] TinyOS Alliance. "TinyOS 2.1 Adding Threads and Memory Protection to TinyOS". In: *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*. SenSys '08. Raleigh, NC, USA: ACM, 2008, pp. 413–414. ISBN: 978-1-59593-990-6. DOI: 10.1145/1460412.1460479. URL: http://doi.acm.org/10.1145/1460412.1460479.

[110] C. L. Liu and James W. Layland. "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment". In: *J. ACM* 20.1 (Jan. 1973), pp. 46–61. ISSN: 0004-5411. DOI: 10.1145/321738.321743. URL: http://doi.acm.org/10.1145/321738.321743.

[111] Nathan Cooprider, Will Archer, Eric Eide, David Gay, and John Regehr. "Efficient Memory Safety for TinyOS". In: *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*. SenSys '07. Sydney, Australia: ACM, 2007, pp. 205–218. ISBN: 978-1-59593-763-6. DOI: 10.1145/1322263.1322283. URL: http://doi.acm.org/10.1145/1322263.1322283.

[112] *Ivy Toolchain*. URL: http://ivy.cs.berkeley.edu/ivywiki/index.php.

[113] Eric Brewer, Jeremy Condit, Bill McCloskey, and Feng Zhou. "Thirty Years is Long Enough: Getting Beyond C". In: *Proceedings of the 10th Conference on Hot Topics in Operating Systems - Volume 10*. HOTOS'05. Santa Fe, NM: USENIX Association, 2005, pp. 14–14. URL: http://dl.acm.org/citation.cfm?id=1251123.1251137.

[114] *TinyOS 2.x Contrib Repository*. URL: http://tinyos.cvs.sourceforge.net/viewvc/tinyos/tinyos-2.x-contrib/.