Wireless network for reliable electric vehicle battery management

Guillaume Le Gall

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'École Nationale Supérieure Mines-Télécom Atlantique Bretagne Pays de la Loire - IMT Atlantique

ÉCOLE DOCTORALE n° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

Guillaume LE GALL

Wireless Network for Reliable Electric Vehicle Battery Management

Thèse présentée et soutenue à IMT Atlantique campus Rennes, le 28 septembre 2021
Unité de recherche : IRISA
Thèse N° : 2021IMTA0258

Rapporteurs avant soutenance :

Fabrice THEOLEYRE	Chargé de Recherche HDR, CNRS, Université de Strasbourg
Valeria LOSCRÌ	Chargée de Recherche HDR, INRIA Lille Nord Europe

Composition du jury :

Président :	Olivier BARAIS	Professeur, Université de Rennes 1
Examinateurs :	Fabrice THEOLEYRE	CR HDR, CNRS, Université de Strasbourg
	Valeria LOSCRÌ	Chargée de Recherche HDR, INRIA Lille Nord Europe
	Xavi VILAJOSANA GUILLEN	Professeur, Universitat Oberta de Catalunya
Dir. de thèse :	Nicolas MONTAVONT	Professeur, IMT Atlantique
Co-encadrant :	Georgios Z. PAPADOPOULOS	Maître de conférences, IMT Atlantique

Invité(s) :

Dominique POISSONNIER	Directeur Recherche Académiques, Texas Instruments Europe

WIRELESS NETWORK FOR RELIABLE ELECTRIC VEHICLE BATTERY MANAGEMENT

GUILLAUME LE GALL

Systèmes Réseaux, Cybersécurité et Droit du numérique
IMT Atlantique

July 2021

SUPERVISORS:
Nicolas Montavont
Georgios Z. Papadopoulos

To my parents and grandparents.

ABSTRACT

The traction battery of an electric vehicle is a key component. It is also a sensitive system for which the voltage and temperature of the cells it is made of must be kept in a given working range. This is the role of the Battery Management System (BMS). The BMS is made of subsystems, called Cells Sensor Units (CSU), which supervise the cells and report their state to a central component named Master Control Unit (MCU). Moreover, they are in charge of performing battery cells balancing, as cells do not have exactly equal capacity, and imbalance between them may appear with usage over time, when they are wired in series. In current BMS implementations, this periodic communication is performed through wires. In this work, we have studied the possibility to replace this wired communication with a wireless medium, by using standardized protocol stack of the Internet of Things (IoT).

After evaluating different communication protocols, we have chosen to base our work on IEEE Std. 802.15.4-2015 Time Slotted Channel Hopping (TSCH). We first have tested this protocol within an battery pack environment, through experimentation, using actual wireless capable nodes. Thus, we were able to determine that the radio links quality is high, that the car's engine electromagnetic emissions should not interfere with the wireless communication, and that most of the problems would come from other users of the 2.4GHz band, and Wi-Fi in particular. We then have sought to determine what the most adapted topology and scheduling management strategies for such a scenario are. To this end, we have proposed two algorithms for centralized network management, based on the Linear Programming and Simple Descent techniques, in order to optimize the topology and slotframe. Considering that many parameters are involved in this optimization work, we have therefore evaluated our algorithms under various setups, and used the results to determine what the best values for these algorithms parameters are. Moreover, we have proposed a routing protocol, which makes use of these algorithms in an iterative way to compute the best possible topology and slotframe, and which allows to propagate the decisions of the centralized network manager to the nodes. This protocol, heavily inspired by the Routing Protocol for Low-Power and Lossy Networks (RPL), relies on periodic messages and asynchronous events to keep the wireless nodes up-to-date with the latest network manager decision. Finally, we have tested this solution with a network of objects in a vehicular environment.

RÉSUMÉ

La batterie de traction du véhicule électrique est un composant clé. C'est aussi un système sensible dont la tension et la température des cellules qui le composent doivent être maintenues dans des plages de fonctionnement bien définies. Garantir cela est le rôle du *Battery Management System* (BMS). Le BMS est composé de sous-systèmes, appelés *Cells Sensor Units* (CSU), qui supervisent les cellules et rapportent leur état à un composant central, le *Master Control Unit* (MCU). Ils ont aussi pour tâche d'effectuer l'équilibrage des cellules, comme celles-ci n'ont pas exactement la même capacité, et du déséquilibre peut apparaître entre-elles au fur et à mesure de leur utilisation, quand elles sont câblées en série. Dans les implémentations de BMS actuelles, cette communication périodique est effectuée de manière filaire. Dans ce travail nous avons étudié la possibilité de remplacer ce réseau de communication par un réseau sans-fil, en utilisant les protocoles standardisés de l'Internet des Objets.

Nous avons évalué les divers protocoles de communication disponibles, et avons choisi de baser nos travaux sur IEEE Std. 802.15.4-2015 *Time Slotted Channel Hopping* (TSCH). Tout d'abord, nous avons testé ce protocole à l'intérieur d'un pack batterie, au travers d'expériences, grâce à des nœuds sans-fil qui l'utilisent. Avec ceci, nous avons été capables de déterminer que la qualité des liens est élevée, que les émissions d'ondes électromagnétiques venant du moteur ne devraient pas impacter le réseau sans-fil, et que la plupart des problèmes qui peuvent surgir viennent des autres utilisateurs de la bande 2,4GHz, et du Wi-Fi en particulier. Nous avons ensuite cherché à déterminer quelles sont les stratégies de gestion de la topologie et d'ordonnancement des transmission qui sont les plus adaptées à un tel scénario. Nous avons proposé deux algorithmes pour une gestion centralisée du réseau, basés sur les techniques de Programmation Linéaire et Simple Descente, afin d'optimiser la topologie et la *slotframe*. De nombreux paramètres sont utilisés dans ce travail d'optimisation, alors nous avons testé nos algorithmes dans un grand nombre de scénarios, et utilisé les résultats pour déterminer quelles seraient les meilleurs valeurs pour ces paramètres. Aussi, nous avons proposé un protocole de routage, utilisant ces algorithmes de manière itérative, pour générer la meilleure topologie et *slotframe* possible, et qui permet de propager les décisions du gestionnaire de réseau centralisé aux nœuds. Ce protocole, largement inspiré de *Routing Protocol for Low-Power and Lossy Networks* (RPL), se base sur des messages périodiques et des événements asynchrones pour garder l'information détenue par les nœuds à jour, sur la base des décisions du gestionnaire de réseau.

Enfin, nous avons testé cette solution avec un réseau d'objets dans un environnement véhiculaire.

ACKNOWLEDGMENTS

My first thoughts here go to Prof. Nicolas Montavont and Dr. Georgios Papadopoulos who offered me to join them in this three years and a half adventure. Thank you for the great learning experience, giving me so much freedom in my everyday tasks, and supporting me in every moment.

I would also like to thank all the people at Texas Instrument who made this collaborative work possible, and in particular Mr. Dominique Poissonnier, Mrs. Xiaolin Lu, and Mr. Jianwei Zhou.

Going through this experience has been great also thanks to all our colleagues at IMT Atlantique. Thank you for all the open discussions, the input you provided, and funny moments.

This work would would not have been that complete without the contribution of the people at Renault, and in particular Mr. Samuel Crégut and Mr. Fabian Ricon-Vija. Thank you for your open state of mind.

Many of the tests and developments presented here have been made possible thanks to the great people at IMT Mines Alès, Mr. Pierre Couturier and Mr. Alexandre Meimouni in particular, thank you for all the great advice, shared knowledge, and laughters.

Thank you to all the students I supervised during this period of time. I hope with me you learned as much as I did.

I would not have reached this point without the support of all the people surrounding me, family and friends. Thank you Iris for your endless support. Thank you Oscar for being such an awesome cat.

All what is presented here has been possible also thanks to the hard work of the whole Free Software community. Among them, I would like to name the Free Software Foundation, thank you for making and maintaining the GNU project, and, most of all, "spreading the word". I would also like to thank the Perl community, for the great programming language that saves so much time in so many situations, and for the sense of humor. Thank you all the people at Framasoft, you really make the Web a better place. And finally, I would like to thank the people making and maintaining the Free Software projects I use every day and love: Debian, Gimp, Inkscape, Blender, Kicad, LibreOffice, LaTex, TexStudio, the classicthesis template, Eclipse, Vim, Kdenlive, and many others.

And to anyone reading this document: happy hacking.

CONTENTS

1 INTRODUCTION

1.1 CONTEXT AND MOTIVATIONS

1.1.1 *Electric Vehicles and Lithium-Ion batteries*

The first electric car in history was made in the 19th century. It rapidly became, together with the internal combustion engine car, an attractive solution to replace horses for moving vehicles. Actually, the first electric car ever built is attributed to french engineer Gustave Trouvé, in 1881. The first car to ever cross the 100 km/h limit, in 1899, was also electric, named the "Jamais contente", visible in Figure 1.1 [1].

But this success has not been enough to push adoption of electric cars forward, and only the cars using internal combustion engine were widely adopted in the course of the 20th century. It is only in the 1990s that a commercial electric car designed for mass production appeared again, the "EV1" made by General Motors. It has been possible thanks to the emergence of the new NiMH battery chemistries. Nowadays, with the new Lithium-Ion batteries, electric cars benefit from a range comparable to the one of those that are propelled by an internal combustion engine, and their adoption increases everyday. They are also considered to be part of the solution to fight climate change [4].

These new Lithium-Ion chemistries, with a nominal voltage around either 3.3V or 3.7V, have a higher energy and weight density than previous existing chemistries. Which makes them a very appealing solution for electrical storage in Electric Vehicles (EVs). However, they are sensitive devices, and for their safe operation, their voltage

1 This image, from an unknown author, is in public domain, available at <https://commons.wikimedia.org/wiki/File:Jamais_contente_parade.jpg?uselang=fr>.

Figure 1.1: The "Jamais contente" electric car, after it crossed the 100km/h limit.

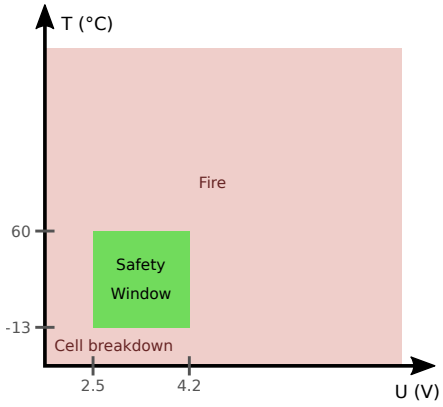


Figure 1.2: Battery cell safety window, based on temperature and voltage acceptable range. Values are given for reference only and may vary depending on cell chemistry and manufacturer.

and temperature must remain in a certain range [5]. If the voltage or temperature would go below the specified value, the cell can be deteriorated. If these values go above the maximum specified, the cell may burn.

For a typical Lithium-Ion cell, the voltage has to remain in the $[2.5; 4.2]$ V range, and the temperature in the $[-13; 60]$ °C range. This range of values is referred to as the cell's safety window. An example of it is shown in Figure 1.2. Please note that the values here are given for reference only, and may vary depending on cell chemistry and manufacturer.

A Lithium-Ion cell, or battery pack, is a complex device, with many non-linearities. While the remaining energy available in the fuel tank of an internal combustion engine car can be easily monitored by measuring a float's position, determining the remaining energy in a battery cell is a complex task. The cell's Open-Circuit Voltage (OCV), which is easy to measure, varies non-linearly with State of Charge (SOC), but also depends on temperature [9]. Furthermore, a cell's voltage will change depending on the charging or discharging current [16]. It is higher as the incoming current increases, and goes down as the outgoing current increases. This phenomenon is called hysteresis, as seen in Figure 1.3 [12]. Please note this figure has been made from generated data with a simulator we implemented, directly based on the equations proposed in [6, 11] and [7], and may not be accurate. The idea here is to show the overall shape of the curve, non-linearities, and hysteresis phenomenon.

These complex phenomena also have an impact on the battery charging process. When charging a battery, first there is a phase in which the charging process is driven in terms of current: the current must not exceed a certain value given by the manufacturer, but also a smaller charging current preserves the battery capacity. This is referred to as the Constant Current phase. Then, when the maximum

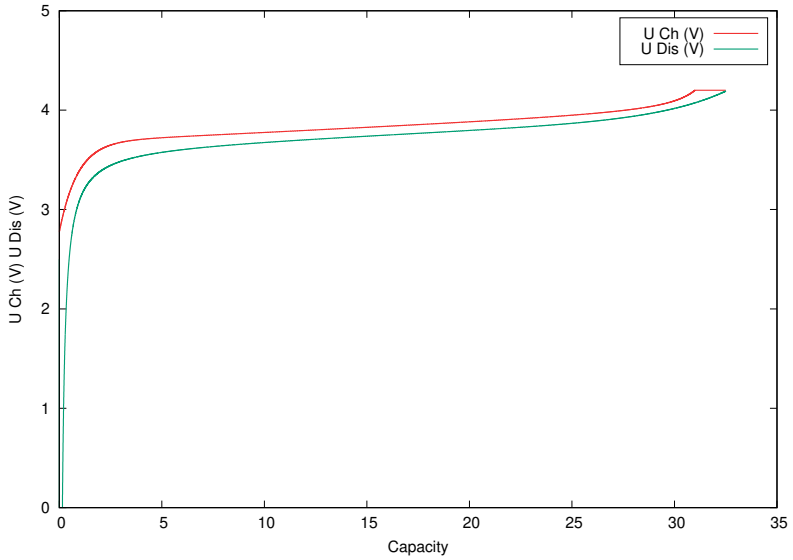


Figure 1.3: Voltage as a function of SOC curve for charging and discharging a battery cell. This plot has been generated from simulated data, and may not be accurate.

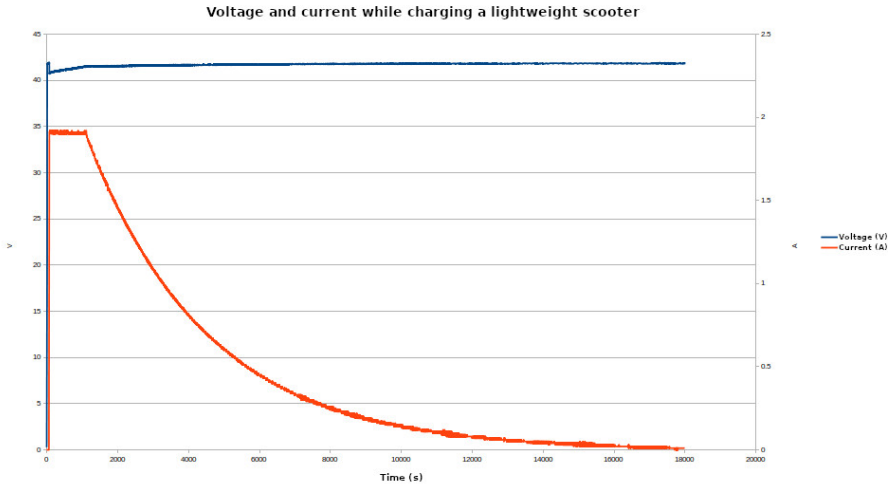


Figure 1.4: Voltage and current curve during the charging process of an electric scooter. This plot comes from an experiment on a real battery. Constant current and constant voltage phases are clearly visible, with the switch happening at nearly 1000 seconds.

voltage of the pack is reached, the charging process is driven based on voltage. This is the constant voltage phase, in which the current decreases slowly. An example of charging a pack with 10 cells in series is provided in Figure 1.4. In this figure, the two phases of the charging process are clearly visible. This plot has been made by recording the voltage and current values during the charging process of a lightweight scooter by our intern, starting at a SOC of around 90% [2].

1.1.2 *Battery Management Systems*

1.1.2.1 *Battery Management System: purposes and implementation*

The Battery Management System (BMS) serves several purposes. First it monitors the individual cell for them to remain in the voltage and temperature range specified by the cell's manufacturer often referred to as the cell's safety window. So the first purpose of the BMS is to make the usage of the pack safer. It does so by allowing the current to flow in and out the pack, either by allowing any other equipment on the Controller Area Network (CAN) bus to draw current from the battery or not, or by controlling a power relay placed on one of the terminals of the battery pack. Should the BMS fail to do so, the battery may be deteriorated if the voltage or temperature goes to low, or burn if one of these parameters goes too high.

It also calculates the SOC of each individual cell [13]. Older SOC calculation methods used OCV together with Coulomb Counting, and newer methods use Kalman Filtering [7]. To provide SOC calculation features, it periodically retrieves the voltage information for each cell. The sampling period of this data depends on the implementation, but the order of magnitude is usually around 500ms [15], [17]. It is safe to assume that the data sampling period will be in the 100ms to 1s range, a lower period meaning a more accurate SOC estimation. If the cell data can not be collected, the BMS can not ensure the safe operation of the pack, and the vehicle should stop to function.

The capacity of each cell slowly decreases over time and with usage. This capacity loss is expressed by a variable called State of Health (SOH), and is also estimated by the BMS.

In a battery pack, the battery cells can be wired both in parallel and in series. When the cells are added in parallel, they tend to balance each other: the most charge cells will naturally discharge in the least charged cell. When they are wired in series, imbalance may appear with usage, so the BMS has to balance them. The cells topology in the pack is often referred to as XS YP, where Y is the number of cells in Parallel, and X is the number of branches which are wired in Series.

2 The results of this experiment and plot have been made available here through the courtesy of Florent Thebaut. Used with permission.

EV Model	Nom. Voltage	Capacity	Max engine power	Number of cells	Pack Weight	Cell capacity
Renault Zoé	400V	41kWh	80kW	96S2P	290kg	214Wh
Nissan Leaf	350V	40kWh	110kW	96S2P	303kg	206Wh
BMW i3 electric	352V	42.2kWh	125kW	96S 1p	208kg e	439Wh
VW eGolf	323V	35.8kWh	100kW	88S3P	—kg	135Wh
Tesla Model S	350V	100kWh	375kW	8 256 (96S 86P e)	625kg	12.1Wh e

Table 1.1: Information available from the Web about some well known electric cars.

For example, 96S1P and 96S2P are popular topologies in commercial electric cars, which give an output voltage between 350 and 400V.

A summary of some well known cars battery topology is presented in Table 1.1, using this terminology[3]. In this table, "e" means this number has been estimated, or deduced, from the information available online. In this table, a voltage of 400V is given for the Renault Zoé. This is most likely the charging voltage of the pack, which is also the maximum voltage the pack can have. Other manufacturers give the nominal voltage of the pack, which is the voltage around 50% SOC. This is also the voltage observed in the portion of the voltage as a function of SOC curve that is relatively flat, as has been presented in Figure 1.3.

Many balancing strategies can be found in the literature. They can be split into two categories [22], [23]:

- Passive balancing: this method aims to dissipate the excess of energy of the most charged cells into resistors. It has the advantage to be easy to implement and cheap, but involves an energy loss. This is the method that is used most often.
- Active balancing: this method most often consists in transferring energy from the most charged cells to a lesser charged cell. It is more energy efficient. However, it involves a greater complexity in the BMS implementation. An active balancing methods may even extend an electric car's range, as it optimizes the battery usage.

A diagram of passive balancing is displayed on Figure 1.5. This circuit is duplicated for each cell in a series. The switch is in fact a transistor, and it is an Integrated Circuit (IC) that controls it with a digital signal, to let the current flow or not. Balancing can be performed

3 Links to the information for each car: Renault Zoé, Nissan Leaf, BMW i3, VW eGolf, Tesla link 1, Tesla link 2, Tesla link 3, Tesla link 4

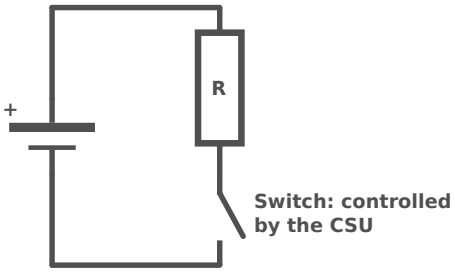


Figure 1.5: Passive balancing of a battery cell: electric diagram.

at any time, but is easier to do when the battery is almost full, as the SOC difference between the cells is more obvious, see Figure 1.3.

Moreover, the BMS will also qualify the overall state of the battery pack and report it (as well as any fault) to the rest of the vehicle through the CAN bus. For example to display this data on the vehicle's dashboard. Depending on how it is implemented, it may also keep track of the evolution of the battery parameters in a log file.

As the number of cells in today's electric car is quite large (usually equal to 96 cells in the series), the BMS is implemented as a distributed system. Its submodules, often referred to as Cell Sensor Unit (CSU) (or "Cells Supervision Unit"), report their data and follow the commands of a Master Control Unit (MCU). These submodules are connected to the MCU through a traditional wired bus (like Serial Peripheral Interface (SPI) or Universal Asynchronous Receiver Transmitter (UART)), usually in a daisy-chain.

Most messages sent from the CSUs to the MCU are cells voltage and temperature data. These are the messages that matter the most, as they allow the MCU to perform its safety oriented tasks, but also to calculate the SOC of each cell, and thus the overall SOC of the pack. Other messages are exchanged over the link between MCU and CSUs. One of them that is important is the balancing command, in which the MCU tells a CSU which cell to balance, in the series it monitors.

A battery pack is organized in modules. Each module is a set of cells, with a CSU to monitor them.

A simplified diagram of a BMS, implemented as a distributed system, is shown in Figure 1.6. Please note that this figure shows the current sensor as part of the BMS, but it may actually be located outside, and its data accessed through CAN.

1.1.2.2 *A closer look at what a CSU is*

A CSU board is designed around an Application Specific Integrated Circuit (ASIC), also sometimes referred to as analog front-end, which will actually do the measurements of voltage and temperature values, using the Analog to Digital Converters (ADCs) contained in its package. It will also perform the actual cell discharge operation by closing a switch that will allow any cell to be discharged into a resistor for pas-

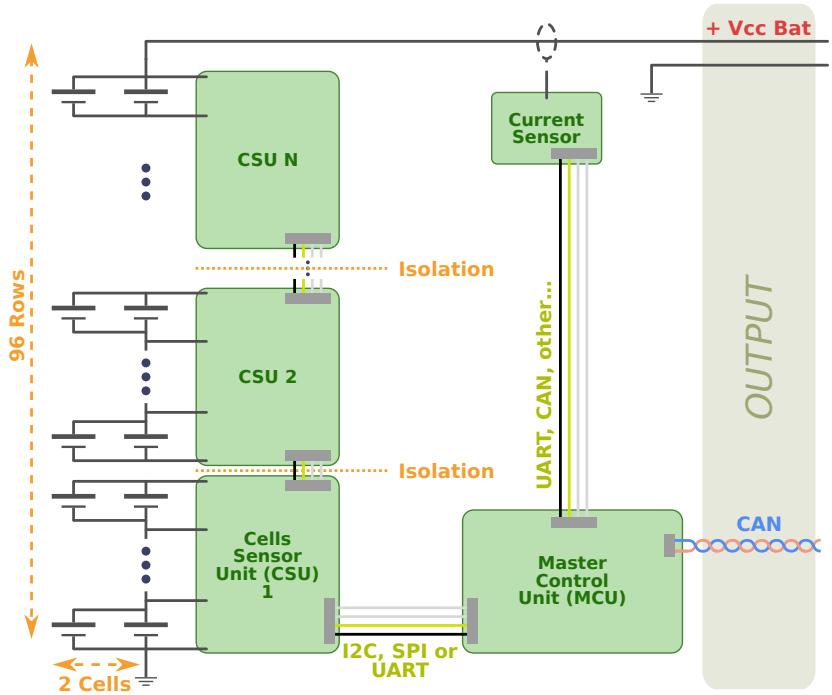


Figure 1.6: BMS simplified diagram, example for a *96S2P* topology.

sive balancing operations, as shown in Figure 1.5. Actually this switch is a Metal Oxide Semiconductor Field Effect Transistor (MOSFET), also contained in the package of the ASIC, which is made conductive upon reception of a balancing command from the MCU. The CSU also has communicating capabilities, as discussed in Section 2.2, to send data to, and receive orders from the MCU. It can also forward messages between the MCU and other nodes, should the device be part of a daisy-chain[4]. A simplified view of a CSU and its features is shown in Figure 1.7. The board represented in the diagram would be the first in the daisy-chain, as it has a direct communication link with the MCU.

1.1.2.3 *A word on complexity, redundancy, and software implementation*

According to [14], as BMS is a safety oriented device, its design must be kept simple, both from hardware and software perspective. Designing a BMS is a long and complex process, and a lot of care has to be taken in selecting the hardware parts that will be used. Moreover, according to the author, the behavior of a BMS software must be deterministic. Especially in a tragic event like a car crash, in which the BMS should have the ability to disconnect the high voltage power rail from the rest of the vehicle. A real time operating system should be used, to make sure the tasks are performed on time. The author makes several important recommendations, like this one: "dynamic memory allocation should be avoided for safety critical embedded systems".

4 For light EVs, like eBikes and small scooters, there is only one CSU, which is located on the same board as the MCU.

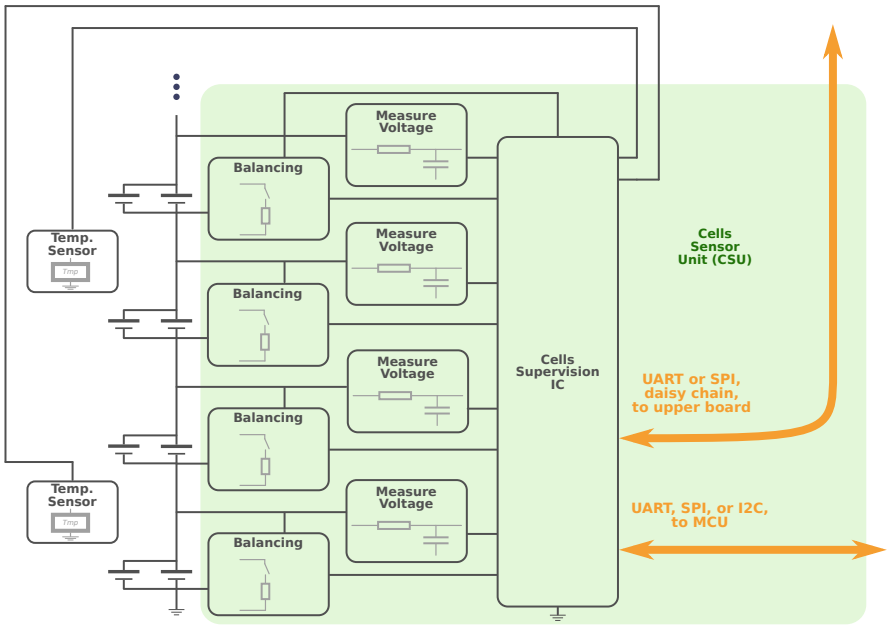


Figure 1.7: CSU simplified diagram, example for monitoring a 4 cells series, and with 2 temperature sensors.

Finally, the software should be kept as simple as possible, to make it easier for developers to debug it, and make problems more obvious. The author summarizes this principle with this phrase, saying the design should be kept "as low as reasonably practicable". Overall, simplicity brings huge upsides in BMS design.

1.1.3 *The IoT stack and well known IoT protocols*

According to what has been discussed so far, there is a necessity to provide communication means between battery subsystems. Looking at what is happening in the fields of digital networks may help to improve them.

The traditional Transmission Control Protocol (TCP) and Internet Protocol version 6 (IPv6) stack have proven their efficiency for developing applications that use a network. These standards have been designed for devices with important computing power, and high datarate links. However, these protocols are often too heavy for embedded systems with constrained links. New protocols have been introduced to allow such objects to use the protocols of the Web. These networks of small, embedded devices, with limited computing capability, constrained links, and often limited energy supply, are referred to as Wireless Sensor Networks (WSN), and the field that studies them is called Internet of Things (IoT).

The IEEE 802.15.4 standard [26], according to its abstract, is for "data communication devices using low-data-rate, low-power, and low-complexity short-range radio frequency transmissions in a wireless

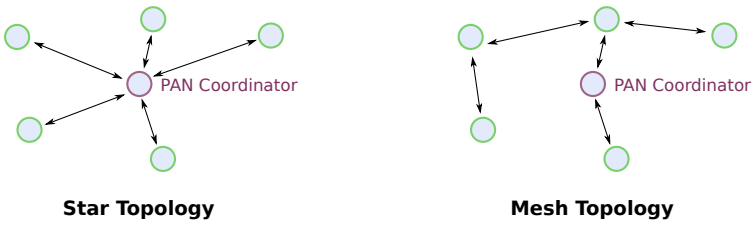


Figure 1.8: Star and mesh topology example.

personal area network". The first version of it was released in 2003. Here, we refer to the 2015 version, that introduced Time Slotted Channel Hopping (TSCH). It is a mechanism that significantly increases reliability, while allowing for parallel transmissions, increasing the throughput in the network. This standard is one of the most appealing solution to build IoT networks.

This standard allows for two kind of topologies:

- Star topology, in which the nodes participating in the network communicate only with the Personal Area Network (PAN) coordinator.
- The peer to peer technology, in which nodes may communicate with each other as well as with the PAN coordinator.

The latter is used to form mesh networks, where the packets can do several hops before reaching their destination. In this case routing is performed by a higher layer. An example of star and mesh topology is presented in Figure 1.8.

In a WSN, the network layer, and the way it works is different from traditional IPv6. To keep the flexibility of the Internet, an adaptation layer is used: IPv6 over Low-Power Wireless Personal Area Networks (6LowPAN) [28]. With this, any node on the Internet may reach a node in the IoT network in a transparent way using IPv6. The role of 6LowPAN is to perform this adaptation while keeping the network layer headers small.

IoT networks may have complex topologies, which may evolve dynamically. They are referred to as mesh networks. When a packet is carried through the network, it must be routed towards its destination. It is the role of the routing protocol to perform this task. And in most cases this protocol is Routing Protocol for Low-Power and Lossy Networks (RPL) [27]. RPL also has self configuration and management features in order to build the topology, based on link layer quality and how they evolve. RPL may also take into account the battery level of the devices, depending on the Objective Function (OF) that is used.

To keep a small size for the packets going through the IoT network, the User Datagram Protocol (UDP) [29] protocol is preferred to TCP. As it is simple, its main upsides are its small header, and small memory footprint in the devices that implement it.

At the application layer, Constrained Application Protocol (CoAP) [30] is the Hypertext Transfer Protocol (HTTP) equivalent for IoT. It is a lighter protocol that reuses the powerful concept of Representational State Transfer (REST) Application Programming Interface (API). In particular, it also has the concept of client and server relationship, with the state of the communication maintained by the client application. It also uses response codes to indicate the kind of response message that is sent, which are again very similar to those in HTTP. The standard specifies that CoAP is used on top of UDP. As this transport layer does not have an acknowledgment and retransmission mechanism, CoAP has its own acknowledgment mechanism to make the exchanges more reliable.

An overview of the Web and IoT protocol stacks are displayed in Figure 1.9.

Application	HTTP
Transport	TCP
Network	IPv6
Link & Physical	Ethernet, Wi-Fi...

Web protocol stack

Application	CoAP
Transport	UDP
Network	RPL
Adaptation	6LowPAN
Link & Physical	IEEE 802.15.4-TSCH

IoT protocol stack

Figure 1.9: Open Systems Interconnection (OSI) model of classical and IoT protocol stacks.

1.1.4 *Appeal for wireless communication in BMS*

Using wireless communication for a BMS application is not the default choice. Copper wires are simple and inexpensive. Using wireless communication links means replacing them with specific microcontrollers that can handle the constraints on Quality of Service (QoS).

Still, there may be upsides in using wireless communication. Removing wires simplifies the assembly procedure, but also saves space and weight, which may have its importance. Having a microcontroller on each CSU also means that the cell data processing can be distributed. Moreover, the information may be stored locally, and reused later, even in the event where the cells are reused in another application. This is known as the "second life" of the battery [24].

In a typical BMS, as the CSUs do not share the same voltage reference (or ground), some additional isolation system must be used on the wired communication bus. This is often provided within the package of the ASIC around which the CSU is built. This extra isolation is not required anymore with wireless communication.

Another key aspect for wireless communication to be attractive is power consumption. The order of magnitude of the power consumed by the wireless devices used for the BMS application has to be similar

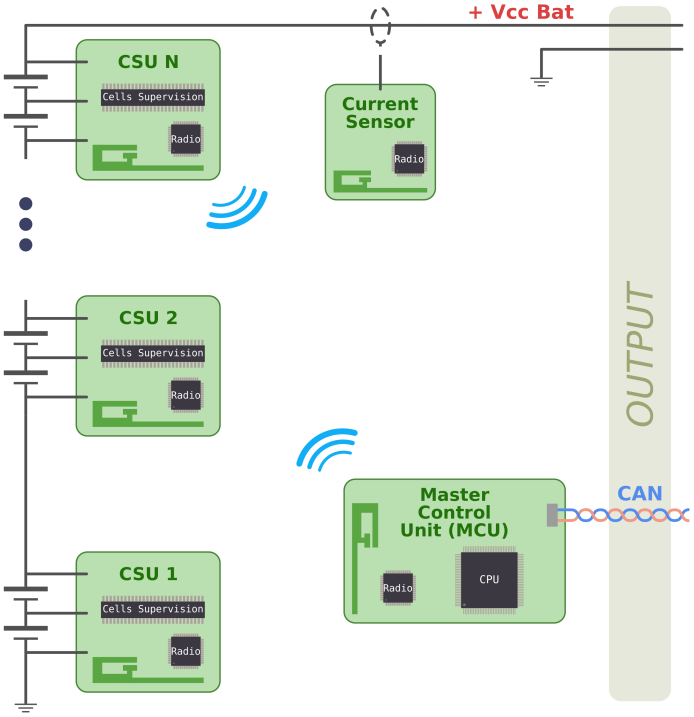


Figure 1.10: BMS with wireless communications concept.

to the power consumed by wired devices, as this change should not have an impact on the vehicle's range. This consideration is further discussed in Sections 2.3.7 and 6.2.2.

The idea that the internal BMS communication wired buses may be replaced by wireless links is shown in Figure 1.10. Note that on this figure the current sensor is also shown as a wireless node, but it may also remain an entity external to the BMS, with traditional wired communication link.

In this design, the CSU would be built around two main chips: still an Analog front-end, to measure the voltage and temperature data, and an embedded Central Processing Unit (CPU) capable of communicating with both the analog front-end and other devices on the wireless network.

1.1.4.1 *Using an IoT stack in the context of a BMS with wireless communication*

In a BMS with wireless communication, most of the application layer packet will contain voltage and temperature data. Any of these values, measured by an ADC within the analog front-end chip on the CSU, is stored in 2 bytes. Also, the sampling period of these values should be small, around 500ms [15], [17].

In a TSCH network, the time is divided into timeslots, which repeat over time, according to a pattern called a slotframe. During a timeslot, a pair of nodes may communicate, which means exchanging a frame

containing data and an acknowledgment. With a typical IEEE 802.15.4 with TSCH mode at the Medium Access Control (MAC) layer, the timeslot is 10ms, which means the data should be transferred to the master node every 50 slots, considering a typical BMS sampling rate. If we decide to dedicate half of these slots to the voltage and temperature collection part of the BMS application traffic, the upper bound in terms of dedicated slots for this in the slotframe is then 25.

As a typical CSU monitors between 6 and 12 cells, and has usually 2 temperature sensors, a CSU data packet is between 16 and 28 bytes. As a typical packet sent over a IEEE 802.15.4 network will have a maximum size of 70 bytes, it is possible to use aggregation at application layer. This is part of what we proposed. Mainly to reduce the number of receiving timeslots at the root node, and leave more space to other packets in the network. This is further discussed in Section 2.4.3.

Although CoAP has many interesting features and allows for flexible applications, it also adds an overhead to the messages of at least 4 bytes [30], while in the case of the voltage and temperature monitoring part of the BMS application the traffic pattern is really simple and well known.

What we suggest to do is to use a custom application packet format on top of UDP for the voltage and temperature monitoring part of the application, and use a CoAP server on the CSU nodes for the other features. A CoAP resource could even be used to turn on and off the sending of the voltage data from the CSUs.

1.2 CONTRIBUTIONS

The purpose of this work is not to determine whether or not BMS with wireless communication is the right approach to manage communications between BMS subsystems. Instead, our goal is to determine if it is feasible, and how is the best way to do it. To this aim, we study wireless network protocols, select the one that seems the more appropriate, and propose ways to optimize network management for such a use case.

The contributions contained in this document are as follow:

STATE OF THE ART As many protocols are available for wireless communication, we provide a state of the art on these technologies, but also the main characteristics of BMS communications as it is implemented today, and the previous efforts that have been done to make BMS with wireless communication possible.

MEASUREMENT OF THE PERFORMANCE OF AN IOT NETWORK INSIDE AN EV PACK To determine what is the impact of an EV environment on a WSN, and the interior of a battery pack in particular, we did several experiments. The goal here was to determine what

would cause packet losses, and what was the influence of position and battery pack enclosure on link Packet Delivery Ratio (PDR).

LINEAR PROGRAMMING APPROACH FOR NETWORK MANAGEMENT Based on the experimental results obtained, we proposed a network management strategy using Linear Programming (LP). It is centralized and based on the knowledge of link quality. The outputs of this technique are not only the IoT network topology, but also the transmissions schedule.

SIMPLE DESCENT APPROACH FOR NETWORK MANAGEMENT The previously mentioned network management method can be improved, especially in terms of computing power and processing time. So we propose a network management method which is more suitable for embedded devices, and appropriate for BMS with wireless communication.

NETWORK PROTOCOL TO PROPAGATE THE DECISIONS As a centralized controller manages the network in the approach we have taken, a protocol is required to propagate the decisions. Our protocol manages the joining procedure, topology and slotframe updates, and includes a few repair mechanisms.

REAL WORLD EXPERIMENT TO EVALUATE THE PROPOSED TECHNIQUES In order to show that the proposed network management strategy and protocol work, we did an experiment with a WSN within a battery pack. The results show that it is efficient, but also that there is room for improvement.

1.3 OUTLINE

This thesis is organized in 6 chapters. In Chapter 2, we present a state of the art on wireless communications for IoT with a focus on BMS with wireless communications. In Chapter 3, we experiment with some IoT networks within an EV battery pack environment. From the results that have been obtained, we elaborate a centralized network management strategy in Chapter 4. As the decisions from the centralized controller need to be propagated to the network nodes, we propose a new mesh network management protocol in Chapter 5. Finally, we give conclusions and perspectives for future works in Chapter 6.

2 STATE OF THE ART

2.1 INTRODUCTION

BMS and IoT networks are complex and deep research topics. Many research papers and articles cover them, but rarely the two together, as BMS with wireless communication is a relatively new topic. In this chapter, we start by studying how data is exchanged in a BMS that uses wired buses for communication between its subsystems. We also provide a review of several layer 2 and above protocols that are available to create a WSN. We also mention other experiments and attempts at creating a BMS with wireless communication. We explain why our choice settled on IEEE Std. 802.15.4-2015 with TSCH. Finally, we describe the methodology used to build this state of the art review.

2.2 A CLOSER LOOK AT THE COMMUNICATION BETWEEN BMS SUBMODULES

The data that is exchanged between the MCU and CSUs, is traditionally carried over a Inter-Integrated Circuit (I2C), SPI, or UART bus. The communication in this link is using a binary protocol, designed by the ASIC manufacturer around which the CSU board is built. The data exchanged is mostly the cell voltage, and temperature. Each sampled voltage or temperature value is encoded over 2 bytes.

The MCU can also trigger balancing for one or more cells at each CSU. The information about which cell should have balancing enabled is usually encoded in a bitmask, where each bit represents the state of a cell.

The size of the frames for these messages will then heavily depend on how many cells are supervised by each CSU.

As said in section 1.1.2.1, the sampling rate of the battery data is dictated by how the MCU has been designed, and has a value of around 500ms.

2.2.1 *Looking at some implementations*

In [18], the author reports on the implementation of a multi-cell, distributed BMS based on some bq76PL536 for the CSUs, and a MSP430 to build the MCU board. In the documentation and code which is provided with the project's report, a few indications on how the MCU communicates with the CSU, and how often, are provided.

For example, the under/over voltage detection, and over temperature detection is configured to be 2 seconds, meaning that if, for example, the CSU detects a voltage that is too high for more than two seconds, it will report an alarm. For data gathering in normal operation, it is triggered by a timer, set to 1 second. The communication between the submodule is request/response, and triggered by the Master, using a SPI bus.

The LTC6803-1 datasheet (from Linear Technology / Analog Devices) also refers to an evaluation interval which can be set between 13ms and 2 seconds for over/under voltage detection [19]. In this document, it is said that reading all ADCs takes 16.4ms. It is also stated there that reading the voltage of all the 12 cells of 3 CSUs, and reports a total time of 472µs for the whole messaging sequence (with a SPI bus speed of 1MHz). In this document, they also describe the messages and their sizes, which are a few bytes long, at most.

A simple BMS can be implemented with an IC providing a reduced set of features, such as the TI bq2947. This device only provides overvoltage detection, which is controlled by a simple timer system. This timer value can be selected between 1 and 2 seconds, by properly selecting the value of a capacitor [20, page 6].

2.2.1.1 *More details with the bq76PL455A-Q1*

More information can be found in the datasheet of the bq76PL455A-Q1 from Texas Instruments. This chip is a typical component to build a CSU board around it, managing up to 16 cells, and is quite recent (first released in 2015). This chip uses a UART daisy chain bus for communication between the MCU and CSUs.

UART messages format The communication protocol with this device is as follows. The MCU always initiates the communication with a request to one or more devices. A request is at least 5 bytes (plus the size of the data) [21, page 50 to 65]. The request frame structure is: frame initialization byte, device/group address byte, register address byte (may be 2 bytes), data byte(s) (usually 1, 5 or 6), and Cyclic Redundancy Check (CRC) bytes (2 bytes). A typical request is thus 6 bytes long. The size of a response frame is similar, with one frame initialization byte, at least one data byte (maximum is 128), and 2 CRC bytes.

Not all requests will trigger a response, this behavior depends on the type of the request. Requests may target a single device, a group of devices, or be broadcast.

The information that can be exchanged with the device is stored in registers, that are 1 to 8 bytes long, big endian. The information available there is about cell voltage, over/under voltage detection, temperature measurement, and fault reporting. Also, some registers

are used to control balancing. Finally, some registers are used to request specific tests within the chip.

How to retrieve the information contained in the input *channels* (meaning the battery cell voltage) is described in page 57, and reports a 6 bytes long request, and a 35 bytes long response. Indeed, retrieving the voltage values of all the channels requires 2 bytes per channel, and there are up to 16 of them. The rest is one frame initialization byte, and 2 CRC bytes.

So if the whole 8 auxiliary channels are requested (*e.g.* containing temperature values), in addition to the 16 "regular" channels, this will be a frame of size $(16+8)*2+3 = 51$ bytes. And thus, if we take a request command such as the one described on page 60 that is 11 bytes long, this is a total of 62 bytes exchanged to request the whole channels values of a CSU.

If we consider a battery pack made of a 96 cells series (a typical number of cells, as can be seen from Table 1.1), it will need 6 CSUs that can handle 16 cells each. So it will require a data exchange of $6*51+11 = 317$ bytes to get the values of all the channels in the whole pack. At the default speed of 250kb/s, and with 10 bits long bytes (see page 30), the time to retrieve the values of all the 144 channels in the pack is:

$$T = \frac{317 \times 10}{250000} = 12.68 \text{ ms} \tag{2.1}$$

ADDITIONAL INFORMATION ABOUT COMMUNICATION SPEED AND DELAY FOR THIS IC This chip uses a UART communication bus, which can be configured: "The baud rate of the communications channel to the microcontroller is set [...] for 125k-250k-500k-1M baud rates. The default rate after a communication reset is 250k." [21, page 30]

When the device is in sleep mode, wakeup requires approximately 1ms [21, page 14].

DAISY CHAIN UART When daisy chain UART is used, all devices in the chain see the packets, that are forwarded through each device in the chain (even the device the packet is not destined to). This bus carries data at 4Mb/s, or 250ns per bit [21, page 33]. The host device still communicates with the device at the bottom of the chain at a maximum of 1Mb/s.

OTHER DELAYS TO CONSIDER Another delay to consider is the sampling time of the ADCs. It generally has a value of 12.6µs, and can go up to 1ms. The measurement can be performed multiple times (from 2 to 32), and the IC can then return the average of the measured values [21, page 26]. This behavior is often called oversampling.

The device also has an "Auto-Monitor" feature, which allows the device to periodically measure data. It can be set between 1ms and 1h.

2.2.1.2 *Conclusions on communication between BMS submodules*

From what has been described in this section, the following ideas can be pointed out:

- The data frames to and from a CSU IC are usually around 10 bytes long, and up to approximately 50 bytes.
- Communication using traditional wired buses is very fast: less than 500µs for a request / response pair, usually around 100 µs, for small messages. An around 12ms are necessary to retrieve the data of a whole battery pack.
- Data acquisition takes less than a millisecond, usually around 10µs.
- Communication on such a wired bus adds very little overhead to application messages.

2.2.2 *CSU board and power supply*

CSU boards manage a subset of cells, packed in a module. But they also need power to operate, and they actually draw their energy from the cells they monitor. Most of the times the ASIC the CSU board is built around embeds a Low-dropout regulator (LDO) voltage regulator [25], or an external (switching or LDO) regulator may be used. As these boards draw current from the cells of the traction battery, their consumption must be kept reasonable, especially when the vehicle is in sleep mode. This means, if wireless communications should be added to CSU boards, the power consumption of this feature must be kept as low as possible, especially when the vehicle is in sleep mode.

2.3 A STATE OF THE ART ON LOW POWER SHORT RANGE WIRELESS TECHNOLOGIES

In this section, we present some existing technologies for wireless communication that could be used for the purpose of this project, or at least that can be used as reference. For all the technologies listed below, some rough information about power consumption is displayed, when available from the literature and has been found. If no power consumption information is displayed, it means none has been found.

2.3.1 *Bluetooth Low Energy*

BLUETOOTH CORE STANDARD Bluetooth is a popular choice for short range wireless communications. Created in the late 1990s, it has been used for many electronic products equipped with a wireless interface. It is a protocol that specifies the way the network should be accessed, from the physical layer to application layer.

BLUETOOTH LOW ENERGY In its Low Energy version, the consumed power is lowered, and it is designed to send short frames in a periodic way. In version 5 of the standard, the physical layer allows a bandwidth of up to 2Mb per second.

Bluetooth Low Energy (BLE) uses the Idustrial, Scientific and Medical (ISM) frequency band, between 2400 and 2480MHz, and splits it into channels of 2MHz width each. So there is no guard frequency between the channels. Three of these channels are reserved for advertising the network.

BLE is simpler to use than Bluetooth, and has been designed to be used in IoT applications. It has interesting features for QoS, including channel hopping [36], as well as channel blacklisting, to temporarily stop using one or more channels for which the QoS is too low [34].

Although BLE allows interoperability with IPv6 networks, and allows mesh networks [38], these features are relatively new, and maybe not as mature as in the IEEE 802.15.4 and RPL stack. Indeed, the BLE Mesh standard from 2017 specifies that the network is a "managed-flood-based mesh network". This means this network does not have an efficient routing mechanism, and nodes only relay packets. They still have some mechanisms to avoid saturation of the network with packets that would be stuck in a routing loop. The standard also says that this routing mechanism could be enhanced in a later version [35].

POWER CONSUMPTION The power consumed by a Bluetooth transceiver while transmitting is 24mW, which is the peak power drawn by the device [37].

2.3.1.1 *IEEE 802.11*

The IEEE 802.11-2016 standards defines a Physical (PHY) and MAC layer for Wireless Local Area Networks (WLAN) [40]. The first version of this standard was released in 1997.

With this technology, the PHY layer can operate in the 900Mhz, 2.4, 3.6, 5 and 60 GHz frequency bands. Initially, only the 2.4 GHz one was used, and with the 802.11a, b, g, n and ac, the 2.4GHz and 5GHz bands are used. Channels are usually 20MHz wide. This means that in the 2.4GHz band, only 4 channels maybe used at the same time with no overlap.

The access to the medium is done through the Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) access method.

The MAC and PHY layer can provide a High-throughput feature, enabling communication at a data rate of 100Mb/s.

MESH ROUTING IN IEEE 802.11 NETWORKS The IEEE 802.11s amendment introduced the mesh networking concept in the standard in 2006 [41]. When this is used, the routing is done by default through the "Hybrid Wireless Mesh Protocol". It is based on both:

- On demand routing, which is mainly based on Radio Metric (RM) Ad hoc On-Demand Distance Vector (AODV) [42].
- Pro-active routing using tree based routing. Which means that if at least one root is present in the network, a distance vector routing tree is built and maintained.

This technique uses a destination sequence number to avoid loops [41, page 43]. When a node needs to communicate with a node outside the mesh, it will generally use pro-active routing, whereas when it needs to communicate with another node inside the mesh, if it does not have an entry in its forwarding table, it will send a broadcast message to request an On-demand path.

POWER CONSUMPTION The power consumed by a IEEE 802.11g or n network node has an order of magnitude of 3 to 5 Watts, the n version being around 50% more efficient than the g version [43].

2.3.2 *Wi-Fi HaLow*

Standardized in IEEE 802.11ah, this protocol has been published in 2017. This version of Wi-Fi, which uses the 900MHz unlicensed frequency band, has been designed to have a reduced energy consumption compared to Wi-Fi, which consumes too much for IoT applications. It also has an extended range compared to the latter. It is actually derived from the IEEE 802.11ac standard, with modified physical and link layers [44, 46].

Wi-Fi HaLow has been designed for point to point communications, in networks using star topology. One of its main advantages is its native support for IPv6. One possible use case for this technology is connecting a IEEE 802.15.4 based network to the Internet [44, 46].

In this technology, the MAC layer uses Enhanced Distributed Channel Access (EDCA), which is derived from CSMA-CA. The protocol, which mostly used for periodic collection of small data samples, also has a feature to rapidly send alert messages, named Restricted Access Window [45].

2.3.3 *WirelessHART*

WirelessHART is a short range wireless protocol for industrial networks, first introduced in 2007, and standardized by IEC 62591 since 2009. The latest version has been released in 2016. It has been made as an evolution of the Highway Addressable Remote Transducer (HART) protocol.

The HART protocol is said to be "hybrid", because it uses digital communications on top of an analog process control link, based on current. This protocol, which exists since the 1980s, standardizes an API as well, which defines how to communicate with a sensor or actuator that implements HART [52].

WirelessHART uses IEEE Std. 802.15.4-2006 for its physical and link layers in the 2.4GHz ISM band. At the MAC layer, it uses both a Time division multiple access (TDMA) and a channel-hopping mechanism. The timeslot duration is set to 10ms. The network manager, an entity located outside the network, is in charge of building a schedule for transmissions, and communicating the appropriate parts of this schedule to the relevant nodes. It is possible to use blacklisting, to stop using the worse channels. This blacklisted channels are configured by the system administrator [48]. It is worth noting that the standard does not specify the channel hopping sequence [49].

At network layer, a mesh topology is used, where every node participating in the network has the ability to route the packets. The routing technique that is used is "Graph Routing", which allows to build redundancy in the routes [51].

The WirelessHART transport layer has optional acknowledgment and fragmentation features [48]. The application layer is HART.

POWER CONSUMPTION The power consumption of a WirelessHART device is 57mW for transmission, and 62mW for reception, according to [53].

2.3.4 *ISA100*

ISA100.11a is a protocol that has been developed by the International Society of Automation (ISA), a consortium based in the USA. Published for the first time in 2009, it is a protocol that has been designed for industrial networks and applications. The standard is now available in IEC 62734 since 2014. It is relatively close and often compared to WirelessHART [48].

Like WirelessHART, ISA100 uses IEEE Std. 802.15.4-2006 for its physical and link layers in the 2.4GHz ISM band, except that it uses a slightly modified version of the MAC layer, making it incompatible with the original IEEE 802.15.4 standard. The link layer uses TDMA as well, together with channel hopping, but with more configuration op-

tions. The timeslot length is set to 10ms by default, but is configurable. For channel hopping, a mode where nodes change channel for each timeslot is available, but there is also a slow hopping mode, in which nodes use the same channel for a period varying from 100 to 400ms. Slow hopping is generally used for contention based periods, in which the nodes try to access the medium on concurrent basis. Finally, it is possible to use hybrid hopping, which alternates between slotted hopping and slow hopping [50]. In ISA100, the system administrator may choose between 5 predefined hopping sequences [49].

Any equipment implementing ISA100 does not necessarily have the ability to advertise the network, or to route packets towards other nodes. This specificity adds more flexibility, but also requires more planning in advance to deploying the network. The network stack is based on 6LowPAN. The routing process uses mesh under in the subnets at link layer, and uses standard IPv6 otherwise. The transport layer is UDP with enhanced integrity check. One of the main differences between ISA100 and WirelessHART is that in the case of ISA100, the standard fully specifies how the backbone of the production site should be implemented [48].

The ISA100 application layer uses two main mechanisms: publication and alerts. The standard also proposes extension which allow each vendor to add more features and adapt to specific customer requirements [47]. The protocol also offers a tunneling feature, which allows to exchange packets of any application layer on top of ISA100. For example it is possible to use HART on top of ISA100 [48].

The criticism that is often made against ISA100 is that the wide flexibility it offers may make objects from different vendors not interoperable, even though they are both complying to the standard [49].

POWER CONSUMPTION According to [54], the average power consumption of a ISA100 device is around 5mW. However, this value probably varies with traffic patterns and the actual bandwidth usage.

2.3.5 *Wireless IO-Link*

Wireless IO-Link is derived from IO-Link, a fieldbus protocol (a standard for wired communication between sensors and actuators, including wiring and a communication protocol) [55].

It uses a master / device (star topology), in the 2.4 GHz band, which is here divided into 80 channels, and uses the Gaussian Frequency Shift Keying (GFSK) modulation. The master can communicate on 5 radio channels at a time, and manage the connection with up to 40 devices. As for a typical IEEE 802.15.4 network, the range is between 10 and 20m.

The traffic pattern is rather simple, as at the beginning of the slotframe, the master sends a multicast frame, then the nodes reply to the

master in a unicast frame. For reliability, the TSCH technique is used, together with a retransmissions mechanism. The standard also offers to use frequency blacklisting.

2.3.6 *IEEE Std. 802.15.4-2015*

2.3.6.1 *Physical layer and medium access methods*

The IEEE 802.15.4 standard [26] is for "short range, low power, low datarate wireless devices". In the first version of the standard, released in 2003, the medium access method was based on Carrier Sense Multiple Access (CSMA), a random access technique. With this method, only one channel is selected, and all nodes in the network use and share it. In the 2015 version of the standard, another technique has been introduced: TSCH.

The standard offers to use several frequency bands, going from 169MHz to 2.4GHz. A frequency band is designated by its central frequency. Each band is divided into channels, and it is possible to find the center frequency of each channel with a formula. For example, in the 2.4GHz band, the frequency of each channel is obtained with:

$$F_c = 2405 + 5\ (k - 11)\ \text{MHz},\ k \in [11; 26] \tag{2.2}$$

In the 2.4GHz band, channels are set every 5MHz. The bandwidth used for the communications is actually of 2MHz, and the remaining bandwidth around the channel, named guard frequency, is left unused to avoid any overlap between neighboring channels.

When the network starts, the coordinator advertises it by periodically broadcasting a frame called beacon. This frame contains information about the network and how to join it, including its identifier, named the PAN ID. A node successfully joining the wireless network may broadcast beacon frames as well, to allow more devices to join. A device trying to join the network will start by scanning all the channels that may be used by the network it is trying to join. Once it receives a beacon, it may initiate the joining procedure.

2.3.6.2 *The TSCH technique for accessing the medium*

The latter relies on two principles: time division in slots, and the usage of multiple channels for a same wireless network.

In this approach, the goal is to reduce the collision risk between packets by sharing and allocating the time and channels, and giving the nodes dedicated periods to transmit, called timeslots. According to the standard, a timeslot is a short period of time which must be long enough to allow two nodes to exchange a frame and an acknowledgment. It is also an opportunity for a node to transmit or receive a packet.

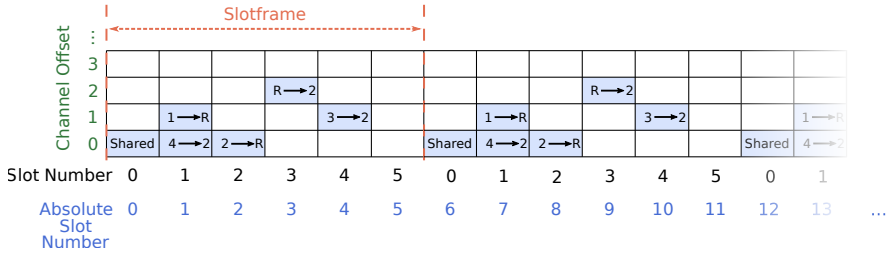


Figure 2.1: Example slotframe, of size 6, for a 4 nodes plus PAN coordinator network using TSCH.

What organizes the communications is the slotframe, which is a pattern containing a schedule telling the nodes when they may transmit or receive packets, and which repeats over time. Each timeslot in the slotframe may be shared or dedicated. If it is shared, any node can use it to transmit, and the potential collisions that may occur are resolved using CSMA in these timeslots. If the timeslot is dedicated, the potential sender and receiver are pre-established, and no other node may transmit in this timeslot. An example of a slotframe is presented in Figure 2.1. The size of the slotframe is configurable, and depends on what the network is deployed for. It is common to have a slotframe size of around a hundred timeslots. Timeslots are usually configured to last for 10 milliseconds each.

All the nodes in the network are synchronized, and switch from channel to channel over time, in order to minimize the effects of interference on a particular timeslot. The goal is to avoid having the same timeslot using always the same weak channel, in every slotframe. This technique is called channel hopping.

In order to achieve this, the nodes count the number of timeslots that have elapsed the network formation. This variable is name Absolute Slot Number (ASN), and is also announced in beacon frames that advertise the network. When a node must transmit or listen for an incoming packet in a given timeslot, it finds the physical channel it must use based on the channel offset and the ASN, using the formula in Equation 2.3.

$$CH = HSL((ASN + ChannelOffset) \% NumChannels) \quad (2.3)$$

In this equation, CH is the physical channel which is going to be used for transmission or reception. The HSL function, which means Hopping Sequence List, is actually a permutation of the channels identifiers. It is either known in advance by all the nodes in the network, or announced in a beacon. With this, it is possible for any node which has joined the network to translate a channel offset and timeslot number into an actual physical channel that it should use. The standard gives a default hopping sequence, but it also offers the possibility to use a custom sequence.

What is not specified in the standard, and is absolutely required for the TSCH network to function properly, is the schedule, which is the actual content of the slotframe. According to the standard, the way to build and distribute the information contained in the schedule is let to "an upper layer".

2.3.6.3 *Minimal requirements to start a network*

The Internet Engineering Task Force (IETF) describes what the minimum configuration and bootstrap procedure for a TSCH network should be in [32]. In particular, the network should offer a "minimal cell" so that new nodes can request to join the network.

When a node has successfully joined, it may use the 6top protocol (6P) to dynamically request cells [33]. However, this IETF draft does not specify what the Scheduling Function should be.

The Section 2.4.4 describes some techniques that can be employed to build a TSCH schedule.

2.3.6.4 *Power consumption*

In the datasheet of the TI CC2650 [82], which is a device capable of using IEEE 802.15.4, we can see that the current consumption rises up to 10mA during transmission periods, while the consumption when the chip is only running code is close to 1mA. Which means if the chip is powered by a 3.3V supply, its power consumption varies between 3.3mW and 33mW.

2.3.7 *Summary on power consumption*

When available, we displayed the power consumption of devices implementing these different technologies. The goal was to give an order of magnitude for this parameter for the technologies presented. To have an extra reference, we can consider the power consumed by a CAN transceiver, which is a common device for wired communication in the automobile industry. In [10], the power consumption of two CAN transceivers is studied. They consume between 23mW (for the chip supplied with 3.3V, when it lets the bus is in a recessive state it draws 7.1mA), and 295mW (for the chip supplied with 5V, when it sets the bus in a dominant state and draws 58.9mA).

Overall, the wireless technologies presented here have a power consumption of a similar order of magnitude, and are all below 100mW even when using their radio, except Wi-Fi, which consumes at least 10 times more than the others. This statement allows us to conclude that Wi-Fi is probably not the right choice for wireless communication for a BMS, even if the nodes are powered by the traction battery pack of the car, which has a huge capacity compared to regular IoT devices batteries. Finally, the CAN transceivers that are referred to above have

a power consumption of the same order of magnitude as the wireless communication technologies presented. In other words, at first sight these wireless technologies do not seem worse than this typical wired technology that CAN is in terms of power consumption.

2.4 RELATED WORKS

2.4.1 *The impact of a battery pack enclosure on a WSN*

In the case of BMS with wireless communication, the network nodes are placed inside an enclosure, often made of metal. Alonso *et al.* [76] consider one CSU per battery cell, which would be placed inside a metallic case. They highlight the harshness of this environment, mentioning that "the presence of reflective objects (mainly metallic battery cell cans and housing) in the vicinity of the antennas, their parameters (impedance, resonance frequency, efficiency, antenna factor) are modified with respect to the ones in free space". They explain that the way antennas will be affected depend on their position within the battery pack. Also they say that the effect of multipath propagation will be important. And that, "two of the three dimensions of the space where the waves are spread (length and width, the height of this space is always very small) are smaller than the wavelengths for some frequency ranges of interest, the environment of the wireless channel behaves as a microwave resonant cavity." Finally, this problem gets even more complex when the battery pack is divided into several blocks.

To measure the possible effects caused by the above described constraints, they build two emulators, a big one ($1.7 \times 1 \times 0.05$ m) and a smaller one ($0.8 \times 0.5 \times 0.05$ m). They used a Vector Network Analyser to get the transfer function of a link between pairs of nodes within this emulator, set at different places. Their results show that at some frequencies the signal is strongly attenuated, and this occurs more often when the frequencies increase.

They conclude saying that Planar Inverted-F Antenna (PIFA) antennas used in the 2 GHz band give the best results in most situations and are suitable for both emulators, and that the helix antennas used at low frequency bands remain an option.

In [74] the same authors carry further experiments with the same evaluation method, and claim the feasibility of in-BMS wireless communications.

In [77] still the same authors tried to further evaluate the choice of the frequencies used, and conclude that working at high frequencies (*i.e.* between 2.2GHz and 2.6GHz) should give better results when the number of nodes increases.

In [78] they explain they developed a simulator which allows them to evaluate the impact of the size of the emulator, and the position of the nodes inside the emulator.

In [79] they enhance their prototype by testing it for communications. They propose their own custom MAC layer, based on TDMA, but report issues when using it, as noise is interpreted as data in their experiment.

2.4.2 *Electro-Magnetic Interference from other systems in the vehicle*

The BMS is often located near other powertrain components, like the engine, which will emit electro-magnetic waves which may disturb the WSN.

However, the frequencies on which the powertrain components emit noise should be different from the frequencies used by a WSN, and the case should act as an electromagnetic shield [74]. Here, the authors describe the thermal noise, and the self generated noise as the most problematic noise sources for the system.

This is also what Revol *et al.* find when they studied the Electromagnetic Interference (EMI) emitted by an AC three phase drive train [75]. The authors studied the EMI below 30 MHz, and find that the peak EMI emission frequencies are found below 100 kHz.

2.4.3 *Choosing a topology for an in-car WSN in which all nodes are in range of each-other*

In [56], the authors study how to efficiently build a wireless network topology to gather data from onboard sensors in a car, where all nodes see each other, which permits a star topology. However, some links can be of poor quality. They make the statement that a two level tree (in which the nodes can be: the root, of rank 1 or of rank 2) can help maximize the path quality towards the root. They use the fact that the carried data is very short and perform data aggregation at the application level in the intermediate nodes.

This technique is particularly suitable for a BMS with wireless communication application, as all nodes are in range of each other, and the application layer data packets are small enough. Thus, this is an idea we reused in our proposition described in Chapter 4.

2.4.4 *Related Work on Topology and TSCH Schedule Management*

KEY PAPERS AND IDEAS There are two approaches to build a mesh network topology: centralized and distributed. The latter, although being highly scalable, relies on local decisions made by the nodes participating in the network, based on partial information. This is the strategy currently implemented in 6TiSCH (and especially in IPv6 RPL [27]).

It is possible to let the nodes negotiate their dedicated timeslots with their neighbors based on the topology decided by the upper

layer routing algorithm, as it is done by the Orchestra scheduling strategy [57].

A centralized controller can also be used to compute the schedule for all the nodes, as is performed in the TASA strategy [58]. This approach has the upside of making decisions based on full information about the network, at the cost of extra communications.

The same authors also proposed DeTAS, a more distributed strategy, where each node computes the number of packets they will have to send per slotframe [59]. In order to make further evaluation of the proposed algorithm, they implemented it on the OpenWSN software platform [60]. For the hardware platform, they used TelosB motes, which use the CC2420 radio chip.

The solutions mentioned here are not directly applicable to communications for battery management, as in this application, given the amount of data to be sent, most of the bandwidth is used, and we do not necessarily want to rely on RPL to maintain the topology, given the small number of nodes and the fact that they are all in range of each other.

OTHER EXISTING SOLUTIONS Many other solutions are available from the literature for building a TSCH schedule. A few of them are listed below.

Morell *et al.* [62] propose to replicate what is done with traffic engineering in the Internet, mainly with Multi-Protocol Label Switching (MPLS). They propose to use time slot numbers as MPLS labels.

Daneels *et al.* [63] propose a "Recurrent Low-Latency Scheduling Function", using Linear Programming to build the schedule.

Hosni *et al.* [64] introduce a method focused on minimizing the end-to-end delay, by allocating consecutive cells for nodes next to each other in the RPL Destination Oriented Direct Acyclic Graph (DODAG). By doing so, they are capable of delivering any packet within the slotframe, disregarding the length of the route.

Soua *et al.* [65] propose "Wave: a Distributed Scheduling Algorithm for Convergecast in IEEE 802.15.4e Networks", which is focused on minimizing the energy consumption, by lowering the duty-cycle of the network.

Chang *et al.* [66] propose "LLSF: Low Latency Scheduling Function for 6TiSCH Networks" a method which is also focused on reducing "the gap" between consecutive slots, according to the topology of the RPL tree. This is similar to [64].

Other possible approaches are listed by Teles Hermeto *et al.* in [67].

2.4.5 *Some BMS with wireless communication implementations in the literature*

Other known attempts at using wireless communication for BMS include a proprietary solution developed by Navitas Solutions Inc. [80], and a proof of concept by Linear Technology, which has been integrated into a production car, using their own wireless communication solution [81].

2.5 PROTOCOLS USED IN THE PROJECT: IEEE 802.15.4-2015 TSCH AND THE IOT STACK

In this project, we focus on short range communications. Also, the CSU nodes will be powered by the battery cells they supervise. So, to choose a wireless technology, there are a few criteria to consider. First the technology should be efficient in terms of energy consumption. It should have or allow some reliability related features. It should allow a reasonable datarate to carry the data. Finally, it should be flexible in terms of link and network layer management.

Wi-Fi, which is standardized by IEEE 802.11, could be interesting, as it is well known and supported, and widely used by several consumer devices. It also provides high bandwidth. However its access method is CSMA, which is not the best suited for layer 2 reliability features. But the main issue with this technology is probably its high power consumption, as described in Section 2.3.1.1, several order of magnitudes above the one of a CAN transceiver [10], which, as a communication technology, is emblematic of wired in-car networks.

This means Wi-Fi is not an option, as it consumes too much energy. Wi-Fi HaLow has been designed for long range communications, which is not what is required here. Industrial protocols like WirelessHART may not provide enough flexibility at application layer or in terms of network management. Wireless IO-Link could be appealing, as it has reliability features built in the standard, but is not flexible with its only one traffic pattern available, and does not allow mesh networking, which seems to be an important feature to have due to the many metallic obstacles in the battery pack. Finally, although BLE is an appealing choice, and despite its support for 6LowPAN [39], it may not be as advanced as IEEE 802.15.4 based networks in terms of topology and routing management, which is why the latter has been chosen in this work. This standard, which has been designed with industrial application requirements in mind [31], seems to be the best choice for our use case.

Upper layer protocols we use are protocols that have proven their efficiency together with IEEE 802.15.4 and TSCH: 6LowPAN and UDP. Although RPL could have been used to manage the topology, we

believe a more efficient solution can be found for our very specific use case, as discussed in Chapter 5.

2.6 METHODOLOGY FOR BUILDING THIS STATE OF THE ART

Some formal methodologies exist to build a state of the art, like "Systematic Literature Review" [83]. They often consist in selecting a large set of papers at a given point in time, and filtering and sorting them with automated and/or manual techniques. They can be a great way to explore the state of the art for a specific domain, as long as one knows what they are looking for, and thus how to build the initial set of papers.

Here, the context was really different. This work is part of not one, but many, research fields, ranging from electrical engineering to network protocols. I knew from the beginning that I would have to learn a lot, and especially to learn what research papers and ideas actually matter for this subject. Furthermore, I knew the subject would demand a lot of time in terms of experimentation and implementation, which gave little time remaining to explore the state of the art of the various domains that relates to this subject. Finally, I knew that the papers that would be helpful would not come to me at once, but all along the way. So I decided that this state of the art would be built in a "best-effort" way of thinking, including to it the papers that appear to be useful at the time they come up to me.

I do not see the state of the art papers we use in such a project as just a list, but rather as a database. What matters with a paper is of course the Portable Document Format (PDF) file with its content, but also some metadata like the date and keywords, its reference in Bibtex format (or whatever other bibliography format is used), and the notes and remarks one can write when reading it.

Great tools can help us manage the paper we discover and that we carry along with us during the whole PhD thesis. Among them are Zotero[1] and Jabref[2]. About Zotero, I understand many people like it, but I personally disliked the interface, and its approach for adding entries to the papers database. Jabref is all about references, and not really about the PDF files themselves, which is not really what I wanted. There are other great pieces of software I could name here.

However, a few months after I became a PhD student, I decided to make my own.

In Unix-like systems, there is this principle that make them very efficient: "everything is a file". So I wanted that the way to add a paper to my database would be by recording a PDF file (or a file in any format) to a specific folder on my hard drive. I wanted to be able to record the updates to this database and share it with other Personal

1 Zotero homepage.
2 Jabref homepage.

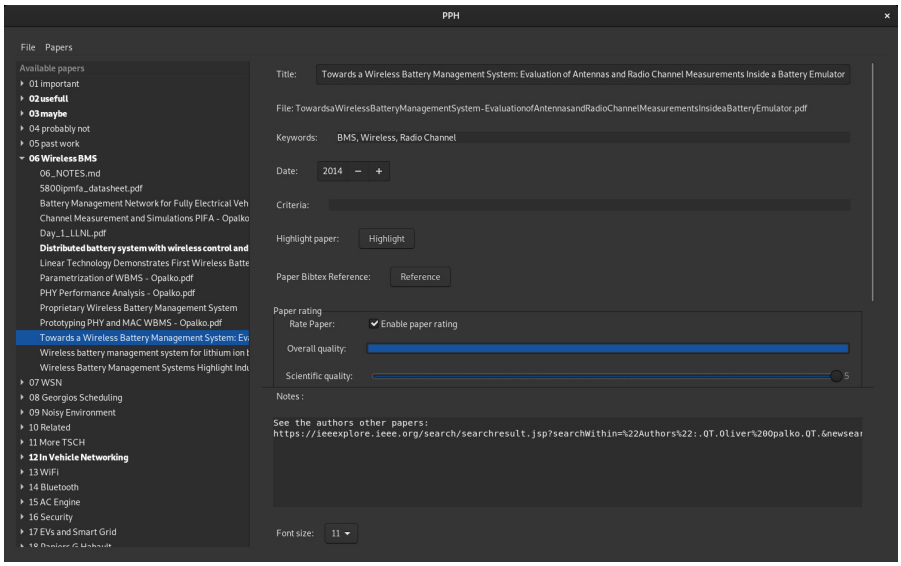


Figure 2.2: Main window of the pph application.

Computer (PC) with Git[3]. Also, I really like the content and layout of the Eclipse Integrated Development Environment (IDE)[4] interface. I like the Gnome desktop environment, so I wanted my software to be well integrated with it. I wanted to have a tight control on everything, especially keywords and the reference for each paper. And finally, I wanted my application to be lightweight and easily extensible, in order to browse and display data from the database I would create.

Based on these ideas, I made a software program I called "PhD Papers Helper", or pph for short. The application is written in Perl 5, is relying on the Gtk3 library for interface management, and on Moose for object oriented programming. The main window of the application is visible in Figure 2.2, and the icon I made for it in Figure 2.4.

The application uses the concept of workspace: actually just a folder, that may contain subfolders, where some files, mostly in PDF format, are stored. The application can handle extra added data about these files that make the database, and store it in specific hidden files in the same folders.

As one can see, the interface is made of a menu bar at the top, and three main panels. The left panel contains the file tree of the workspace. The top right hand panel contains information the user can set manually, like title, keywords, reference, but also rate the paper, on its scientific content, writing quality, and level of difficulty. About the rating, the idea was to be able to easily filter through papers, to quickly find these I would like to share with students (for example because they were well written, or not too difficult to understand). The bottom right hand corner is a space for the user to take notes.

3 Git homepage.
4 Eclipse project homepage.

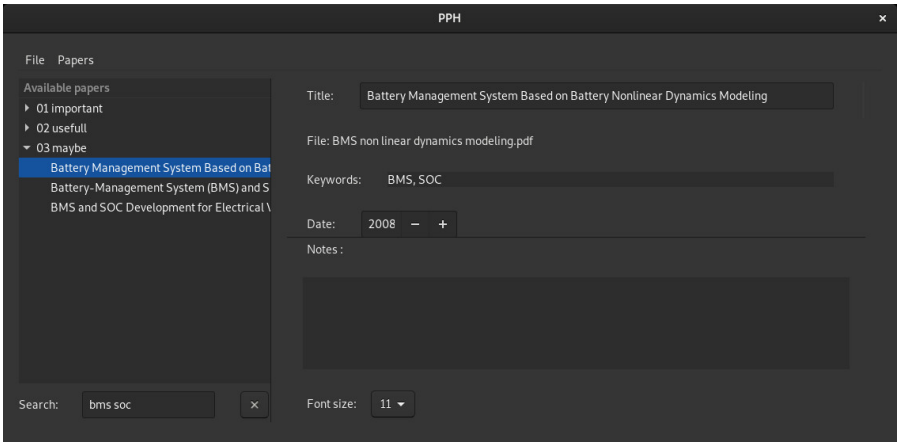


Figure 2.3: pph main window while using the filter feature, with keywords "BMS" and "SOC".

Aside from this, the application has a filter feature, that lets the user easily hide the papers in the file tree that are not related to the words entered in the search bar. The input of this text field is case insensitive. This is where the manual definition of the keywords in the top right-hand panel is handy, as the search feature takes them into account, and it is thus possible to use them to "group" papers, or at least make them appear under a specific keyword, even if the authors did not list that keyword in their paper. An example of filtering is displayed in Figure 2.3.

The program also has a refresh feature, to rebuild the file tree in the left hand panel, which is useful when a paper is added and the application is open. And of course a save feature.

Finally it has a plot feature, relying on the Perl wrapper library for GNU plot. This is here because, during the first year, my supervisors asked me to plot what optimization criteria papers on a specific subject were considering. So instead of just making the plot, I made a feature. The plot dialog of `pph` will consider only the papers that appear in the file tree if a filter is currently being applied. The dialog window for creating a plot is presented in Figure 2.5. And the output diagram it produces with the set of research papers I have in my database is presented in Figure 2.6.

To be able to use it easily on several PC, and share it, I added the `debian` folder to its source folder, to be able to create a Debian package for it.

Of course, as of today, this application is far from perfect, and is unfinished. It has known issues and missing features. I really wanted, but never found the time, to implement an advanced filter feature that works on dates, and rating, but also a feature that fills in the title, date and keywords of a paper automatically when providing the Bibtex entry. Also, I do not know if it is currently portable to anything that is not a GNU/Linux Operating System (OS). Letting aside these

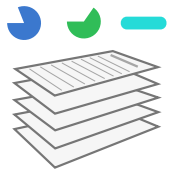


Figure 2.4: Icon of the pph application.

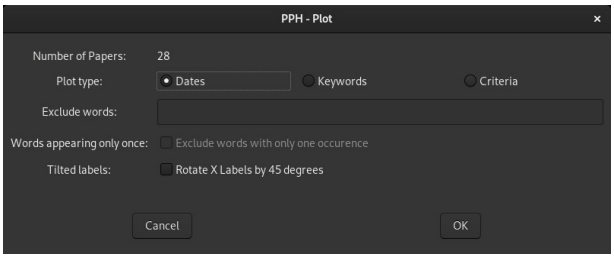


Figure 2.5: Plot settings window for pph. Filter text contains "TSCH", and the plot will be about dates.

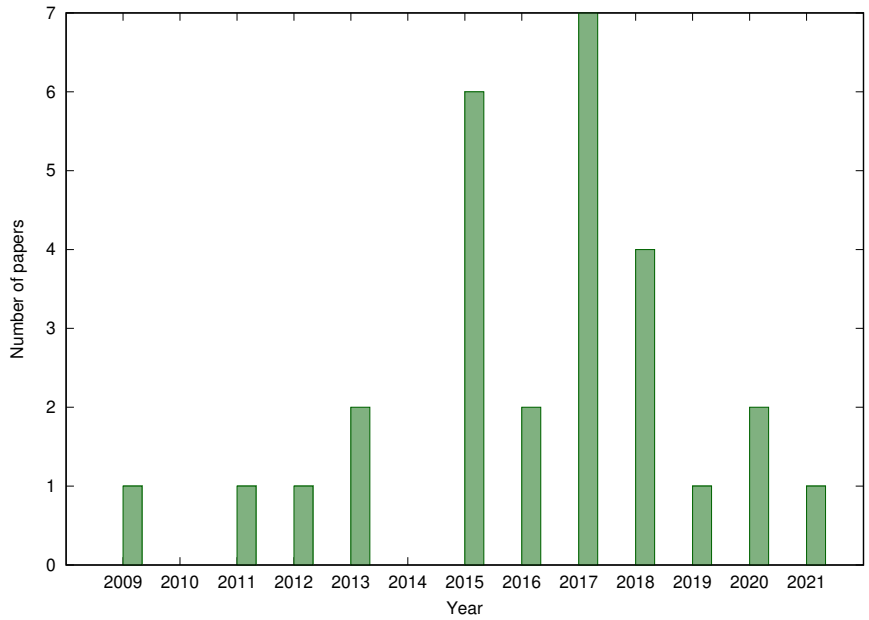


Figure 2.6: Plot generated with pph. Years in which the papers about TSCH in my database have been published.

issues, I really liked having this application to work with, which has been working better than I hoped, and been way more useful than I anticipated. If I would be facing the same choice I had, a little more that three years ago, when starting to build a state of the art database, I would still make the same. And use the same programming language.

The source code of this program is available for download[5], under the GNU GPLv3 or later license. To clone the project on your computer, use the command shown in Listing 2.1.

```
git clone https://redmine.telecom-bretagne.eu/git/pph
```

Listing 2.1: Git clone command for pph.

2.7 CONCLUSION

In this chapter, we presented a state of the art on the research domains that relate to our subject. First, we extended the information provided in Chapter 1 by giving more details about how a BMS works, and how the communication between its submodules is performed in today's architectures. Then, we presented many MAC layer and above protocols that are widely used in IoT applications. Moreover, we also referenced research works that study wireless networks for in-car communications, and BMS with wireless communication in particular. We concluded this study saying we would use IEEE Std. 802.15.4-2015 with TSCH in the rest of our work. Finally, we explained the state of mind and ideas that conducted how this state of the art has been built, and how I made a software program to sort and organize the research papers I worked on.

The ideas and results contained in the research papers we mentioned have been used as building blocks for the subsequent work presented here. The battery pack environment has specificities and may make the wireless network behave differently than it would in an open-air environment, as discussed in Section 2.4.1. This is why we study how an IEEE Std. 802.15.4-2015 TSCH network performs within a vehicle's battery pack environment in Chapter 3.

5 pph Web link: <https://redmine.telecom-bretagne.eu/projects/pph/repository>

3

IOT NETWORK PERFORMANCE INSIDE AN EV BATTERY PACK

3.1 INTRODUCTION

Although IoT nodes placed in a battery pack would be very close to each other, it is not obvious that the link quality between them will be very high. The metallic parts of the pack may have an impact on communication, and it is possible that in some situations they prevent two nodes from being in direct sight of each-other. In order to get an accurate idea of what the network performance in an EV battery pack can be, and what is the real impact of battery cells and pack enclosure on wireless communication, we performed several tests on multiple EV platforms. We also tried to evaluate what is the impact of Wi-Fi communication in the environment on the IoT network, and whether or not driving the car influences the network link quality.

Contribution

This chapter presents the following contributions:

1. We perform link quality measurement of a IEEE 802.15.4 TSCH network in various EV platforms, and present the results.
2. We study the impact of node position, Wi-Fi interference, and driving the car, on network link quality.

3.2 HARDWARE, SOFTWARE, AND TESTING PROCEDURES

To get an idea of the base QoS in an EV environment, and more particularly in a battery pack, several tests have been performed.

These tests were done with I3Mote CC2650 boards, and Launchpad CC2652 with battery booster pack boards. The boards are shown in Figure 3.1. They both rely on a PIFA antenna to perform wireless communication.

To limit the risks when doing the tests, the boards were always placed in a plastic isolating case, which are shown in Figure 3.2.

For the tests, the boards were running a firmware developed by Texas Instrument (TI), which implements an IoT stack based on IEEE 802.15.4. On the Launchpad CC2652, it is also possible to use the BLE

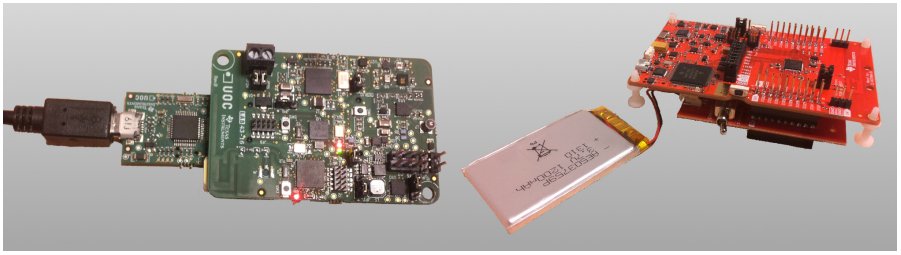


Figure 3.1: I3Mote CC2650 board with daughter board to connect it with Universal Serial Bus (USB) (on the left), and Launchpad CC2652 with battery booster pack (on the right).

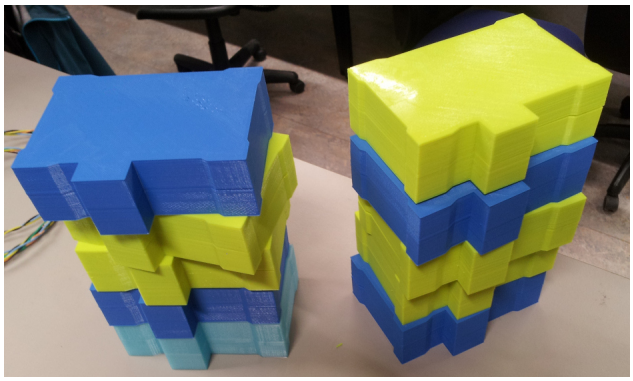


Figure 3.2: 3D printed plastic isolating case for Launchpad boards.

physical layer with a datarate of 2Mb/s. In this implementation the topology is managed with RPL, and a centralized scheduler allocates one receive and one transmit timeslot per slotframe between any node and its parent. The nodes run a CoAP server. One of the resources on this sever allows to retrieve the MAC layer PDR statistics.

The software program used to do the measurements is one we developed ourselves, running on the PC connected to the border router. It performs CoAP GET requests to the nodes periodically, in order to generate traffic. Actually, the request to a particular node is sent upon reception of a response from that same node, or after a timer expires. The program records the latency for each request / response pair, and if the application layer packet was lost. With the software running on the Launchpad CC2652, the traffic pattern is different, as it is programmed to send UDP packets to the root node. Our test management program also regularly retrieves the MAC layer data from each node and logs it (for both hardware platforms).

In these tests, one of the goals was to evaluate link quality with regard to the node positions. So the networks were forced to use a star topology.

One of the platform we used for the tests is a Renault Fluence battery pack. It contains a series of 96 Lithium-Ion battery cells, and has a nominal voltage of 360V. To limit the risk when working with it, it is necessary to have proper protection equipment. In our case we

Figure 3.3: Protection equipment worn to setup the tests in the Fluence battery pack.

used a helmet, gloves, and specific clothes and shoes, to be sure we were safe, as shown in Figure 3.3.

3.3 METRICS USED

When doing these tests, the most important metric we measured was the PDR at the MAC layer. We also often measured the end to end latency and the application layer PDR. These two metrics are of a lesser importance, because when the MAC PDR and the retransmission mechanisms are known, it is possible to estimate them, whereas the MAC PDR depends directly on the link quality, and the environment.

The jitter (i.e. the variance in the latency) could also have been a metric to consider, but it will be more a consequence of the network management choices that are made than a way to qualify the links and wireless environment quality.

To get a better overview of the channel conditions in an EV environment, we place the nodes at several positions in each test, to see how a specific position affects the PDR. We also tried to evaluate the difference between two different physical layers, the default one from IEEE 802.15.4, and the 2Mb/s BLE one. We evaluated the impact of the metallic battery enclosure by doing tests where it was removed from the pack, and compared the results to when it was in place. Finally we did some tests while other devices were using the 2.4GHz ISM band, mainly Wi-Fi devices. This was to see how these devices would interfere with a BMS IoT network.

Figure 3.4: Network topology: testing range in a one hop scenario

3.4 TESTS RESULTS

3.4.1 *Outside the EV environment*

3.4.1.1 *Range test with I3Mote CC2650*

A summary of this test is presented in Table 3.1. The topology is shown in Figure 3.4, and the results can be found in Figure 3.5 and 3.6. This is one of the first test that we have done, and sadly we did not record the MAC layer PDR here. The number of MAC layer retransmissions was set to 3.

Purpose:	Get a preliminary and reference result for application layer PDR, to be compared with results in an EV environment. See the impact of distance on PDR, while having line of sight.
Platform:	None
Date:	12/2018
Hardware:	I3Mote CC2650
MAC/PHY layer:	IEEE 802.15.4-TSCH, with retransmissions
Scheduler and Schedule:	Default scheduler, 127 slots slotframe.
Test time / ending condition:	50 packets per node.
Traffic pattern:	Polling based (periodic GET requests).

Table 3.1: Range test of I3Mote CC2650 summary

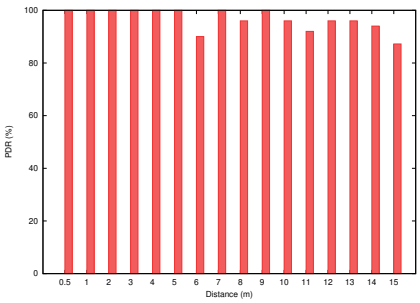


Figure 3.5: Application layer PDR: Range test in a corridor.

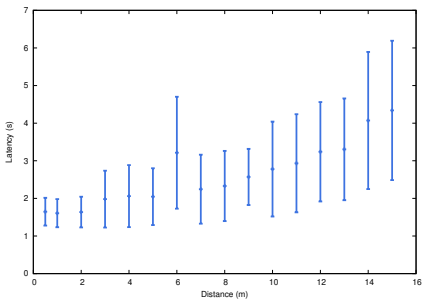


Figure 3.6: Application Latency: Range test in a corridor.

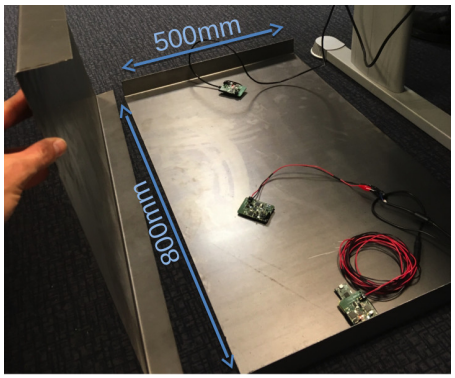


Figure 3.7: Testing channel conditions inside a metallic box.

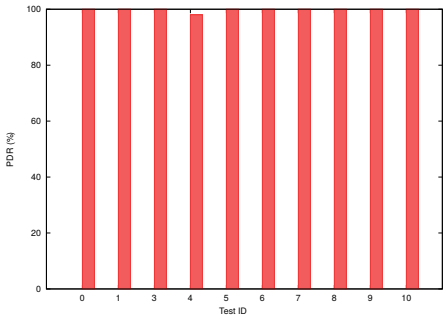


Figure 3.8: Application layer PDR: test in the metallic box.

Test Conclusion: The application layer PDR decreases with distance, more significantly above a 10m distance between the two nodes.

3.4.1.2 *In a metallic box*

We built a box as described in Section 2.4.1, like Alonso *et al.* did and presented in [76], in order to evaluate how nodes would perform in it. We also compared the difference between inside and outside the box.

Purpose:	Evaluate the impact of the enclosure.
Platform:	Metallic box
Date:	01/2019
Hardware:	I3Mote CC2650
MAC/PHY layer:	IEEE 802.15.4-TSCH, with retransmissions
Scheduler and Schedule:	Default scheduler, 127 slots slotframe.
Test time / ending condition:	100 packets per node.
Traffic pattern:	Polling based (periodic GET requests).

Table 3.2: I3Mote CC2650 performance in a metallic box test summary

The result is presented in Figure 3.8. For all the tests that we did, the PDR was almost always 100%, even when the root node was inside, and the other node outside.

Test Conclusion: The results obtained here show a very good link quality, and do not seem to reflect what has been presented in [76]. However, we measured only application layer PDR. More interesting results, focusing on MAC layer PDR, are presented in following sections.

3.4.2 *PGO e-Hemera (at IMT Mines Alès)*

The PGO company, *IMT Mines Alès* and *IUT de Nîmes* have built an electric car prototype. As in most electric cars, the pack is located in the floor. It is a 32S1P pack, with a size of 1.25m per 1.47m, and can be seen in Figure 3.9. During the test, the battery was mounted onto the car, as it would be for normal operation.

During the two tests presented below, the car was parked. Summaries of the tests are presented in Table 3.3 and 3.4. The nodes layout is presented in Figure 3.10 and 3.12. As opposed to what can be seen in these pictures, the pack was mounted onto the car during the test. The results can be found in Figure 3.11 and in Figures 3.13 to 3.20.

3.4.2.1 *First iteration of this test, with I3Mote CC2650*

Purpose:	Measure the impact of node position on MAC PDR inside an EV battery pack. The pack was designed and assembled by IMT Mines Alès. The characteristics of this pack is very similar to what can be found in a medium-sized commercial car.
Platform:	PGO e-Hemera
Date:	03/2019
Hardware:	I3Mote CC2650
MAC/PHY layer:	IEEE 802.15.4-TSCH, with retransmissions
Scheduler and Schedule:	Default scheduler, 43 slots slotframe.
Test time / ending condition:	1500 packets per node.
Traffic pattern:	Polling based (periodic GET requests).

Table 3.3: I3Mote CC2650 performance in the PGO e-Hemera test summary

Test conclusion: We can observe that the MAC PDR is a little bit lower than in the metallic box. This is probably because the nodes are not on top of the battery cells, but are embedded within the pack, often having no line of sight between them and the root. However, we can observe that the average MAC PDR is 99.4% and above, whatever the position of the node is.

To get a better understanding of what the PDR actually is, the total number of transmitted packets from all the non-root nodes is 15502,

Figure 3.9: The PGO e-Hemera at IMT Mines Alès. Hood and trunk cover have been removed.

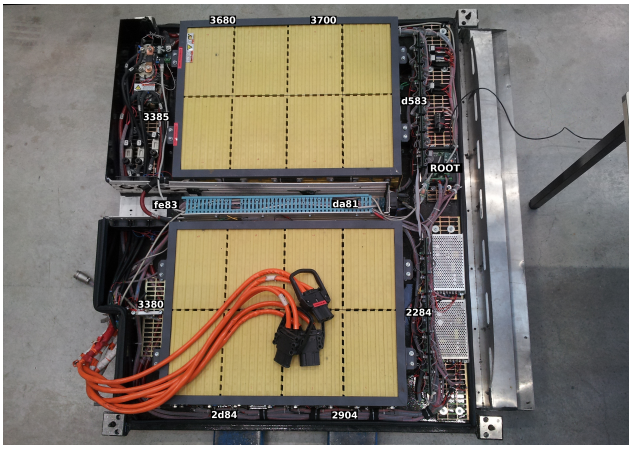


Figure 3.10: Motes layout in the PGO e-Hemera battery pack at IMT Mines Alès.

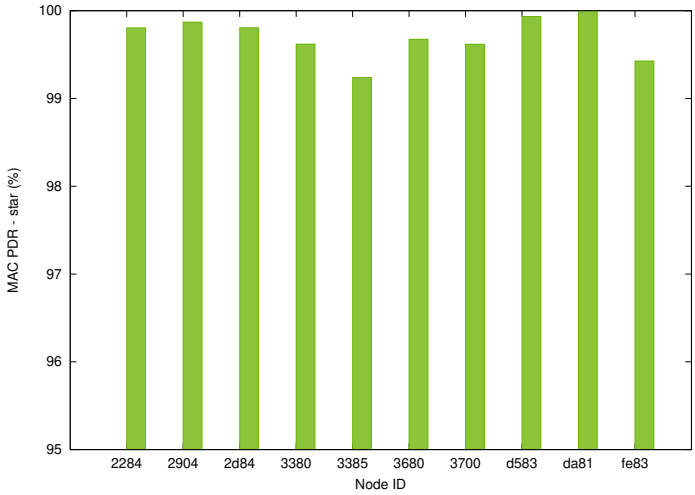


Figure 3.11: Average MAC PDR inside the battery pack (PGO e-Hemera).

and the total number of lost packets is 47 for these same nodes. For example on the graph of the above figure, node "d583" lost 1 packet out of 1529.

3.4.2.2 *Second iteration of this test with Launchpad CC2652*

Purpose:	Measure the per channel MAC layer PDR statistics inside an EV battery pack, with TSCH over the BLE physical layer. The car is parked.
Platform:	PGO e-Hemera
Date:	12/2019
Hardware:	Launchpad CC2652
MAC/PHY layer:	IEEE 802.15.4-TSCH over BLE PHY
Scheduler and Schedule:	Default scheduler, 30 slots slotframe.
Test time / ending condition:	Approximately 30 minutes.
Traffic pattern:	BMS app. over UDP

Table 3.4: Launchpad CC2652 performance in a battery pack test summary.

Test conclusion: As for the test with the IEEE 802.15.4 MAC layer, the PDR is very high. The per channel results show that the losses tend to occur on channels that are close to each other, as presented in Figure 3.19. Here, almost all of the losses occur on channels that overlap with Wi-Fi channel 11. So they may be due to interference.

3.4.3 *Renault Fluence battery pack*

Among all the platforms we used to do ours tests, the one we have used the most is a Renault Fluence battery pack, which can be seen in Figure 3.21. A summary of this test is presented in Table 3.5. The position of the nodes during the test can be seen in Figure 3.22. The measured MAC layer PDR is displayed in Figures 3.23 to 3.28.

Results:

- ID = 2284, PDR = 99.94%
- ID = 2904, PDR = 99.98%
- ID = 2d84, PDR = 100%
- ID = 3380, PDR = 99.98%
- ID = 3680, PDR = 98.71%

Test conclusion: The results are very similar to those obtained in the pack at IMT Mine Alès. The worst average MAC PDR is 98.7%,

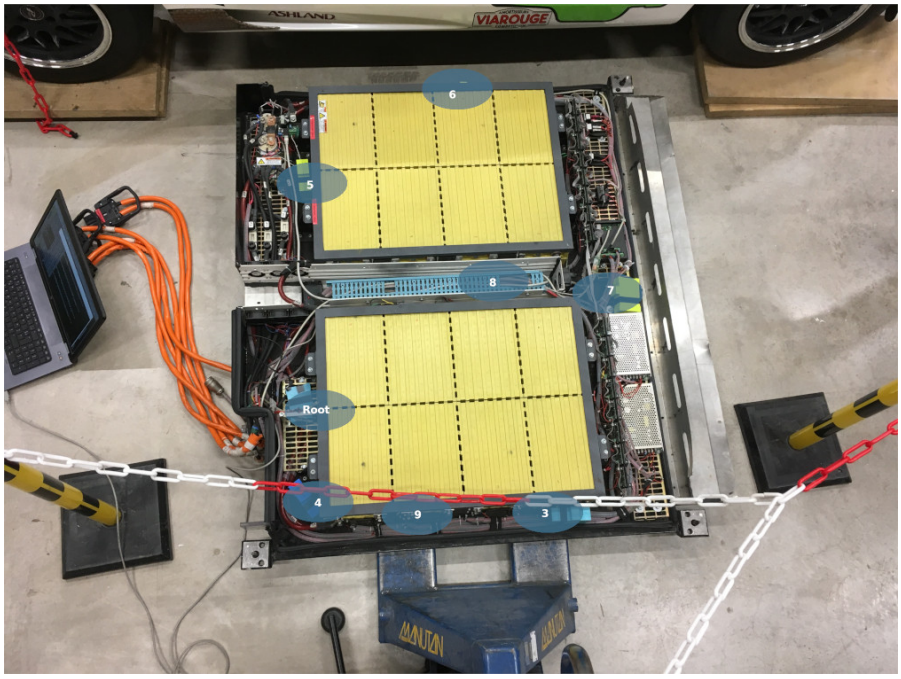


Figure 3.12: Launchpad CC2652 motes layout in the PGO e-Hemera battery pack at IMT Mines Alès.

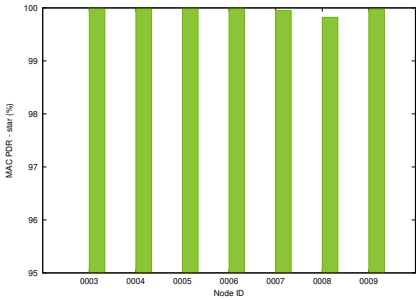


Figure 3.13: Average MAC layer PDR, PGO, inside the pack, parked.

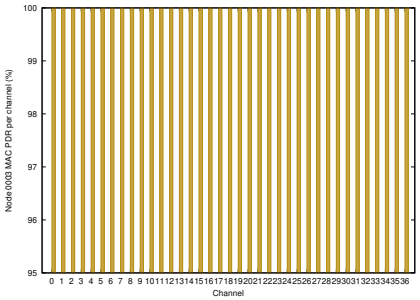


Figure 3.14: Node 3 MAC layer PDR, PGO, inside the pack, parked.

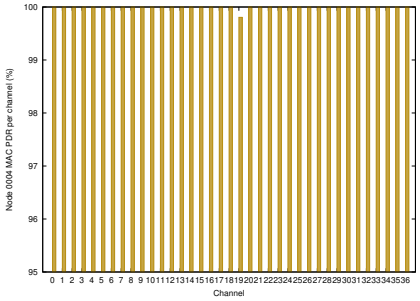


Figure 3.15: Node 4 MAC layer PDR, PGO, inside the pack, parked.

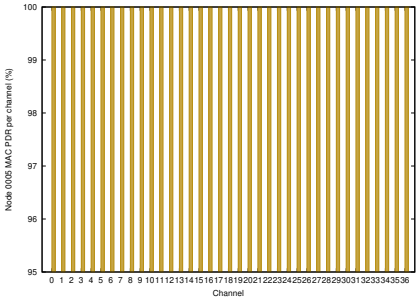


Figure 3.16: Node 5 MAC layer PDR, PGO, inside the pack, parked.

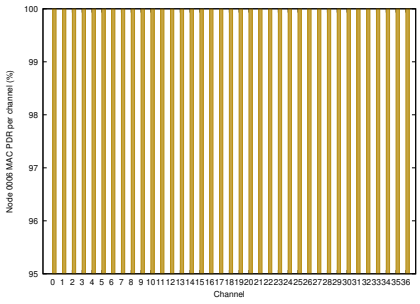


Figure 3.17: Node 6 MAC layer PDR, PGO, inside the pack, parked.

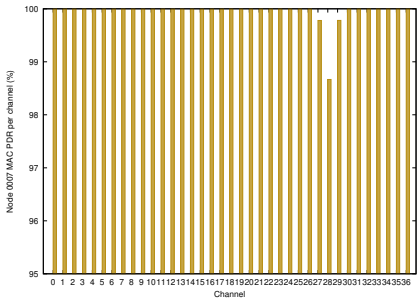


Figure 3.18: Node 7 MAC layer PDR, PGO, inside the pack, parked.

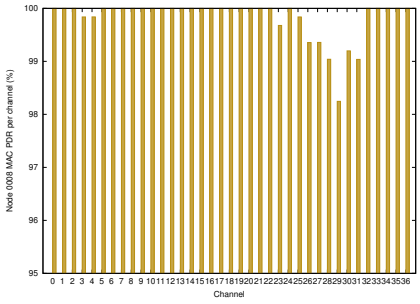


Figure 3.19: Node 8 MAC layer PDR, PGO, inside the pack, parked.

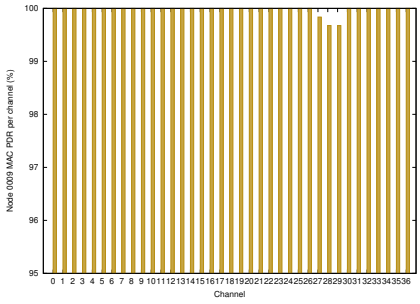


Figure 3.20: Node 9 MAC layer PDR, PGO, inside the pack, parked.

Figure 3.21: Measuring link quality inside a Fluence pack: during the test.

(a) Front view.

(b) Back view.

Figure 3.22: Layout of the motes inside the Renault Fluence pack.

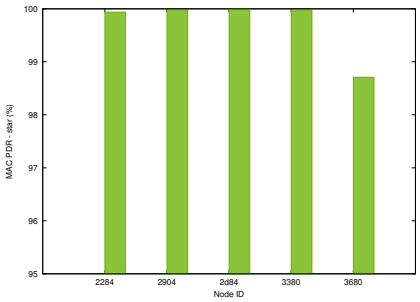


Figure 3.23: Average MAC layer PDR, inside the Fluence pack.

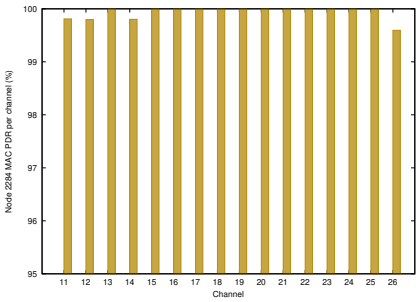


Figure 3.24: Node 2284 MAC layer PDR, inside the Fluence pack.

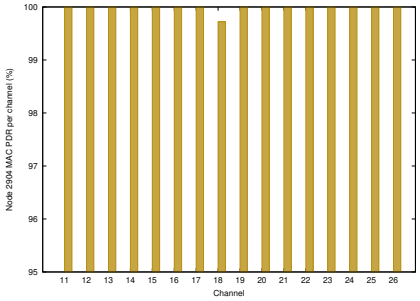


Figure 3.25: Node 2904 MAC layer PDR, inside the Fluence pack.

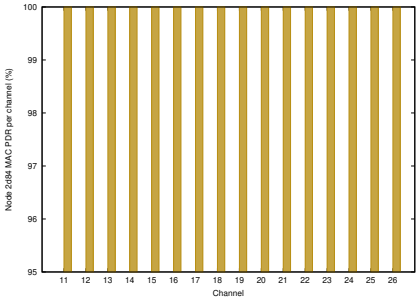


Figure 3.26: Node 2d84 MAC layer PDR, inside the Fluence pack.

Purpose:	Measure the per channel MAC layer PDR statistics in an EV battery pack. This pack is from a commercial sedan, designed by Renault.
Platform:	Renault Fluence pack
Date:	10/2019
Hardware:	I3Mote CC2650
MAC/PHY layer:	IEEE 802.15.4-TSCH, with retransmissions
Scheduler and Schedule:	Default scheduler, 43 slots slotframe.
Test time / ending condition:	2 hours (resulting in 5728 application packets per node on average).
Traffic pattern:	Polling based (periodic GET requests).

Table 3.5: I3Mote CC2650 performance in the Renault Fluence battery pack test summary

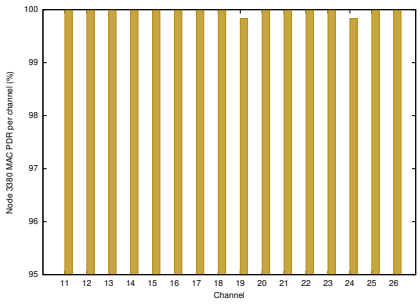


Figure 3.27: Node 3380 MAC layer PDR, inside the Fluence pack.

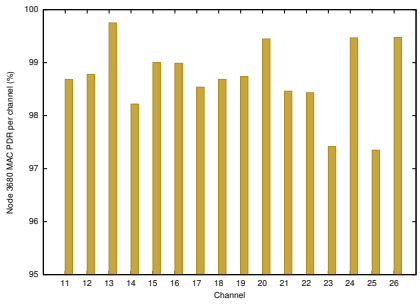


Figure 3.28: Node 3680 MAC layer PDR, inside the Fluence pack.

and all the other nodes have significantly better PDR. For one node the average MAC PDR is even equal to 100%. We can conclude that globally the performance is high, but that more packets are lost in some of the tested positions inside the pack. The per channel data does not reveal that some channels offer less performance than others yet, this result does not allow to conclude on what makes a channel weaker.

3.4.4 *Link quality and interference with Wi-Fi communication*

Another test was performed to measure the per radio channel PDR statistics in the Fluence battery pack. One of the goals was to compare the results inside, and outside the enclosure, also the impact of having the enclosure put on top of the pack or not, and measure the impact of Wi-Fi communication on the IoT network link quality. The layout of the nodes for this test is visible in Figure 3.30. During the first test, the metallic enclosure was removed, and during the second one, it was

Figure 3.29: Link quality measurement inside the Fluence battery pack with continuous Wi-Fi communication in the environment using the IPerf software.

on top of the pack. Both tests were run for one hour, which gives an average of 4100 MAC layer frames sent per node.

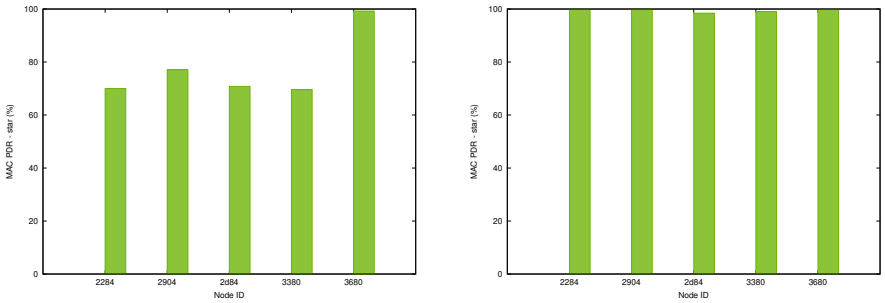
As a reminder, IEEE Std. 802.15.4-TSCH and Wi-Fi are two technologies that use the 2.4GHz band, and some of their radio channels overlap. The most commonly used channels in Wi-Fi are 1, 6 and 11. They overlap with the channels 11, 12, 13, channels 17, 18, 19 and channels 22, 23, 24 of IEEE Std. 802.15.4, respectively.

Both tests were run with Wi-Fi communications on channels 1, 6 and 11. The traffic was generated with the IPerf 2 software, using a total of 5 PCs: 3 running an IPerf client, and 2 configured as access point and running an IPerf server. One of the 2 access points had 2 Wi-Fi cards, which were configured to operate an access point on two different channels with these 2 cards. The actual setup used in these tests is shown in Figure 3.29. Summaries of these tests are displayed in Table 3.6, and Table 3.7.

Figure 3.31 shows the average MAC layer PDR for each node. Figures 3.32, 3.33, and 3.34 show the results for the per radio channel MAC layer PDR for nodes 2284, 2904 and 2d84, which were placed inside the pack. Results for the two other nodes, 3380 and 3680, are not displayed here but are very similar to the other three.

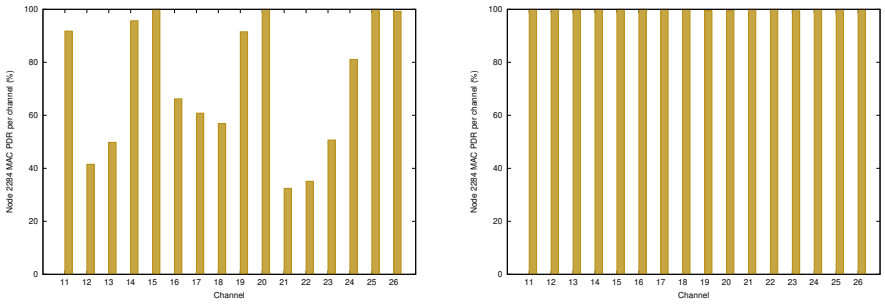
Test conclusion: These results mean the enclosure makes the wireless communication better inside the pack and, to some extents, it protects the network from external interference. However it is hard to tell how much the communication improves with the enclosure on top: in a previous version of the test, for which the results are not displayed here, the enclosure was not properly put back, and Wi-Fi communication had a stronger impact on the IoT network.

Figure 3.30: Measuring link quality inside a Fluence pack with Wi-Fi interference: motes layout



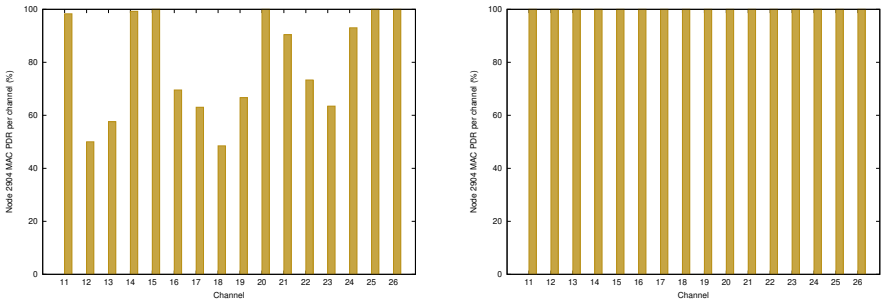
(a) Without pack enclosure (b) With pack enclosure

Figure 3.31: Average MAC layer PDR for the Fluence pack with Wi-Fi traffic in the environment



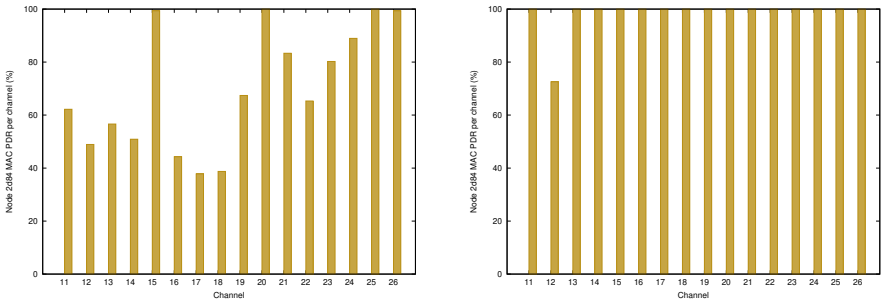
(a) Without pack enclosure (b) With pack enclosure

Figure 3.32: Node 2284 MAC layer PDR for the Fluence pack with Wi-Fi traffic in the environment



(a) Without pack enclosure (b) With pack enclosure

Figure 3.33: Node 2904 MAC layer PDR for the Fluence pack with Wi-Fi traffic in the environment



(a) Without pack enclosure (b) With pack enclosure

Figure 3.34: Node 2d84 MAC layer PDR for the Fluence pack with Wi-Fi traffic in the environment

Purpose:	Measure the impact of Wi-Fi interference on MAC PDR.
Platform:	Renault Fluence pack
Date:	02/2019
Hardware:	I3Mote CC2650
MAC/PHY layer:	IEEE 802.15.4 TSCH
End condition:	1 hour
Additional setup:	No pack enclosure, Wi-Fi communications added.
Result:	MAC PDR >= 69.6%

Table 3.6: MAC PDR in a Renault Fluence battery pack test summary, without pack enclosure, and with Wi-Fi traffic on channels 1, 6 and 11 in the environment

Purpose:	Measure the impact of Wi-Fi interference on MAC PDR
Platform:	Renault Fluence pack
Date:	02/2019
Hardware:	I3Mote CC2650
End condition:	1 hour
Additional setup:	With pack enclosure, Wi-Fi communications added
Result:	MAC PDR >= 98.4%

Table 3.7: MAC PDR in a Renault Fluence battery pack test summary, with pack enclosure, and with Wi-Fi traffic on channels 1, 6 and 11 in the environment

Finally, please note that the traffic pattern used for Wi-Fi is not realistic, as packets were sent continuously with the IPerf 2 software to occupy the radio channels. In a real world scenario, with less traffic on Wi-Fi channels, and with the metallic body of the car on top of the pack, the interference could even be less significant.

3.4.5 *Measuring the impact of driving the car*

Here the goal is to try to see if driving the EV would have any impact on a BMS with wireless communication. Two tests have been performed. One in *IMT Atlantique* in Rennes, on a platform named Open-Source Vehicle (OSV). And the other was done at *IMT Mines Alès*. Summaries of these tests are available in Table 3.8 and 3.9. The results can be found in Figures 3.38 to 3.49, and Figures 3.51 to 3.58.

3.4.5.1 *First test for measuring the impact of driving with OSV*

The OSV is a car chassis that was sold by an Italian company of the same name. It is designed as a lightweight urban vehicle. Its battery

Purpose:	See the difference in PDR at the MAC layer between still and driving vehicle. For the "still" test, the vehicle is parked inside the IMT Atlantique building.
Platform:	OSV
Date:	11/2019
Hardware:	I3Mote CC2650
MAC/PHY layer:	IEEE 802.15.4-TSCH, with retransmissions
Scheduler and Schedule:	Default scheduler, 43 slots slotframe.
Test time / ending condition:	45 minutes for each test (equivalent to approximately 2000 packets per node)
Traffic pattern:	Polling based (periodic GET requests).

Table 3.8: I3Mote CC2650 performance on a moving EV chassis test summary.

Figure 3.35: The OSV inside *IMT Atlantique*.

pack is made of a series of 24 $LiFePO_4$ cells, and its engine has a power of 19kW. A picture of it is visible in Figure 3.35, and the placement of a node during the test is shown in Figure 3.36. The layout of the nodes during the test can be found in Figure 3.36.

During this test, the OSV was being driven outdoor, in the schoolyard. As discussed below, the Wi-Fi access points of the school, and the devices using them, may have caused interference and may have had a significant impact on the results. During the whole test, the car was accelerating and braking, which means it did not have a constant speed, and its average speed was probably around 20km/h. These are more or less the conditions that can be found while driving in a city center (both in terms of vehicle speed and 2.4GHz band usage).

For the presentation of the results, the numbers on the Figures, on top of each bar, represent the actual number of packet lost during the test. Please note that the number of packets transmitted per node varies between the nodes. The results for the parked vehicle are shown

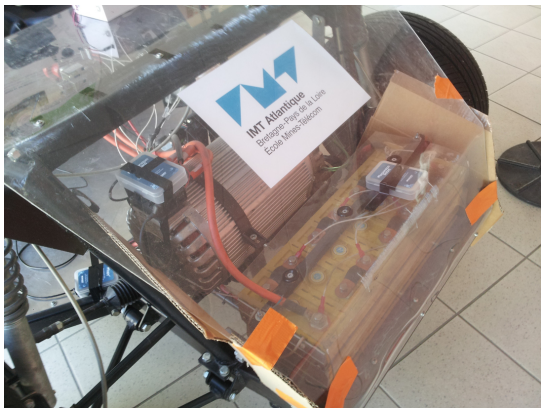
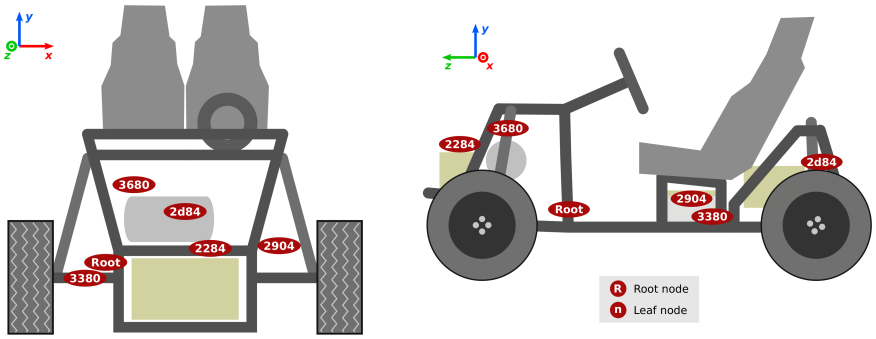


Figure 3.36: Mote placement on the front of the OSV for the drive test.



(a) Front view.

(b) Side view.

Figure 3.37: Motes layout on OSV

on the left-hand side, and for the driving vehicle on the right-hand side.

Test conclusions: The results of this test are not trivial to read and analyze. Although at the first look it seems that the number of packet lost increased outside while driving, the results there are actually better in the latter (252 packets lost inside, 192 while driving outside).

To properly interpret these results, it is important to remember that Wi-Fi devices were present and being used in the environment during both tests. And the channels 1 and 6 of Wi-Fi overlap with channels 12 and 13, and 17 and 18 of IEEE 802.15.4, respectively.

Node 2284 was placed in the front of the car, on top of the front part of the battery pack. It was not in sight of the root node, due to the engine being between the two nodes. It suffers from losses mainly on channels 12, 13, 17 and 18: the channels that overlap the most with Wi-Fi. If only the other channels are considered, there is only a subtle decrease in PDR (PDR_{parked} = 100% whereas $PDR_{driving}$ = 99.70%). And all the losses except one occur on channels that are still overlapping with Wi-Fi channels 1 and 6 (IEEE 802.15.4 channels 12, 16, 19).

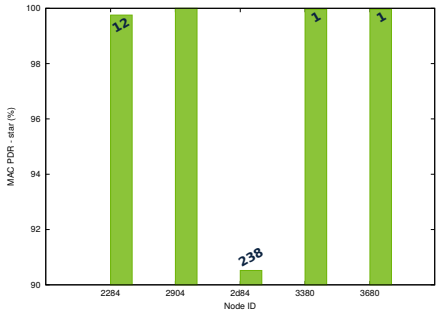


Figure 3.38: Average MAC layer PDR, OSV, parked.

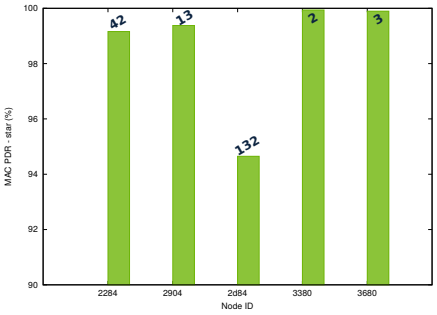


Figure 3.39: Average MAC layer PDR, OSV, driving.

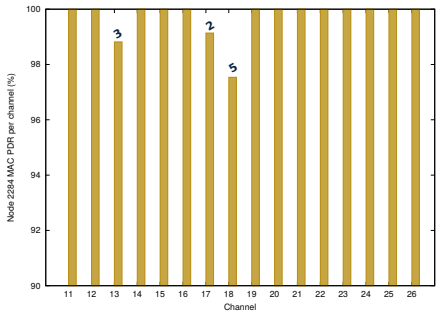


Figure 3.40: Node 2284 MAC layer PDR, OSV, parked.

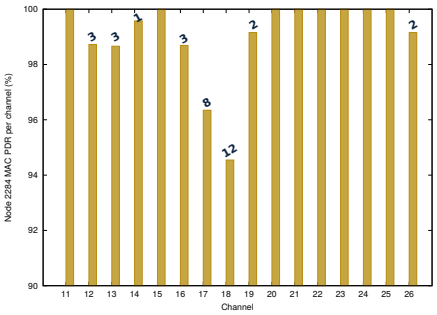


Figure 3.41: Node 2284 MAC layer PDR, OSV, driving.

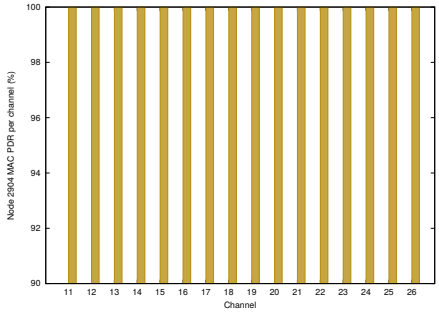


Figure 3.42: Node 2904 MAC layer PDR, OSV, parked.

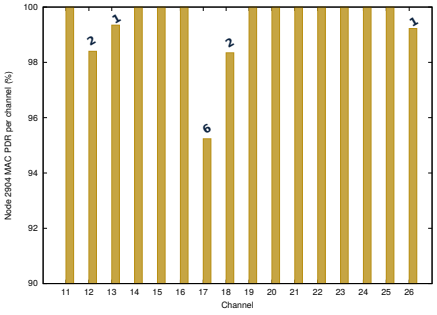


Figure 3.43: Node 2904 MAC layer PDR, OSV, driving.

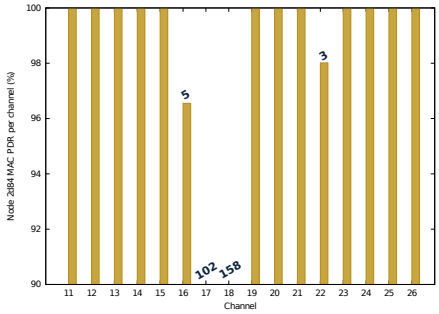


Figure 3.44: Node 2d84 MAC layer PDR, OSV, parked.

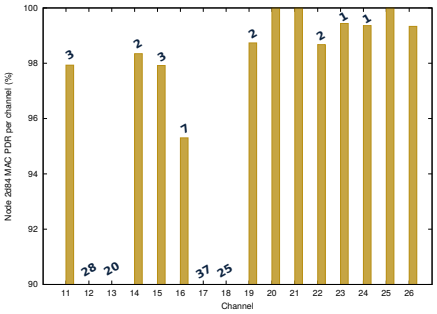


Figure 3.45: Node 2d84 MAC layer PDR, OSV, driving.

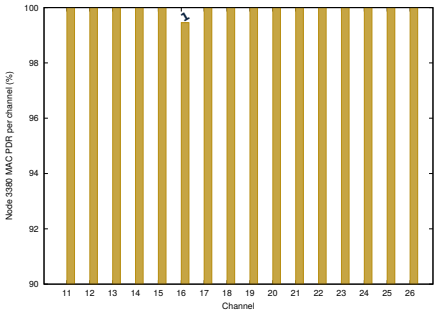


Figure 3.46: Node 3380 MAC layer PDR, OSV, parked.

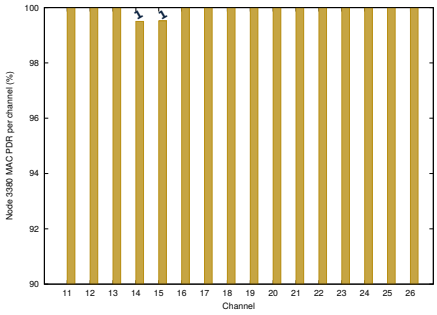


Figure 3.47: Node 3380 MAC layer PDR, OSV, driving.

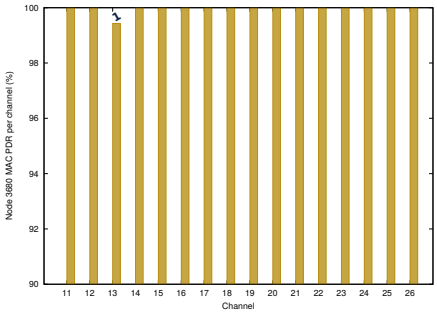


Figure 3.48: Node 3680 MAC layer PDR, OSV, parked.

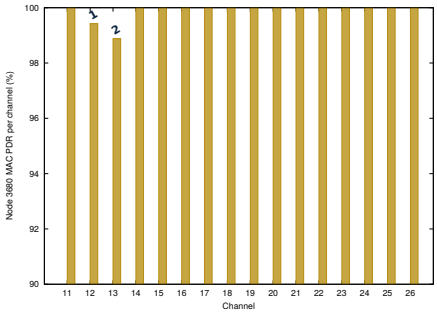


Figure 3.49: Node 3680 MAC layer PDR, OSV, driving.

Node 2904 was placed in the center left of the car, on top of the BMS and under the driver's seat. It shows very good results on the channels that do not overlap with Wi-Fi, despite being more than a meter away and not in clear sight of the root node.

Node 2d84 was placed at the back of the car, on top of the battery charger. Among all the nodes, it was the furthest and with no line of sight with the root node. It is the node that suffers the most of losses on the channels overlapping with Wi-Fi. It is also the node with the most significant drop while driving on the non overlapping channels.

Node 3380 was placed in the center right of the car, right behind the passenger seat. It was placed relatively close to the root node, but with no line of sight. Overall it is the node with the best performance considering the two situations, with a PDR almost equal to 100%. Node 3680 was placed at the front of the car, on top of the engine and in sight of the root node. It has very good performance while both parked and driving, with losses only in the channels overlapping with Wi-Fi.

Overall it is possible to conclude that the devices of the powertrain of the car do not interfere with the wireless communication in the 2.4GHz band, or at least not in a significant manner. This is consistent with the information available from the literature. Furthermore, the channels that overlap with Wi-Fi channels are severely disturbed, in both situations. The nodes that perform best are the one that was very

close to the Root, despite being on top of the engine, and the nodes that were between the seats and the floor (but still close to power electronic devices). The node at the back was the furthest from the root, but also the more visible, with no metallic cover from the chassis at all. It is hard to tell if the extra losses it encountered while driving were due to the fact that the car was actually moving, or if it was just more exposed to the environment. And the second hypothesis seems to better comply with the behavior measured at the other nodes.

3.4.5.2 *Second test for measuring the impact of driving with PGO e-Hemera at IMT Mines Alès*

Purpose:	Measure the per channel MAC layer PDR statistics near a driving EV engine, with TSCH over the BLE physical layer.
Platform:	PGO e-Hemera
Date:	12/2019
Hardware:	Launchpad CC2652
MAC/PHY layer:	IEEE 802.15.4-TSCH over BLE PHY
Scheduler and Schedule:	Default scheduler, 30 slots slotframe.
Test time / ending condition:	Approximately 30 minutes, preformed 2 times, one with the car parked, and one with the car being driven.
Traffic pattern:	BMS app. over UDP

Table 3.9: Launchpad CC2652 performance in a driving EV test summary.

The non-root nodes were outside the pack during this test. The root node is placed inside the battery pack. Our intention was initially to the test with the nodes inside the battery pack. But their batteries were depleted before we started. As removing the battery from the car, or putting it back, are operations that take one hour, it was not possible to access the nodes inside the pack to charge their batteries. Instead, we decided to attach some nodes near the engines.

During this test, the car was being driven outdoor, around the campus of IMT Mines Alès. The Wi-Fi access points of the school, and the devices using them, may have caused interference and may have had a significant impact on the results. During the whole test, the car was accelerating and braking, with a speed varying probably between 10 and 40km/h. The car being driven during this test can be seen in Figure 3.59.

Test conclusion: The results are slightly different from what has been obtained with the OSV. The PDR is slightly worse when the car is being driven. But as the root node was in the battery pack, and the other nodes were near the engines, this setup is relatively different from what would happen in a BMS with wireless communication.

Figure 3.50: Mote placement at the back of the PGO, near the engine, for the drive test.

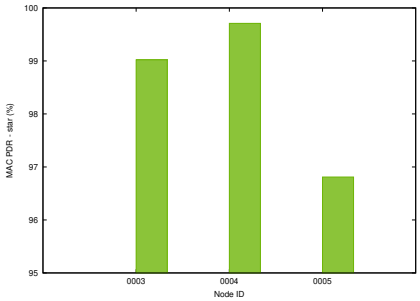


Figure 3.51: Average MAC layer PDR, PGO at IMT Mines Alès, parked.

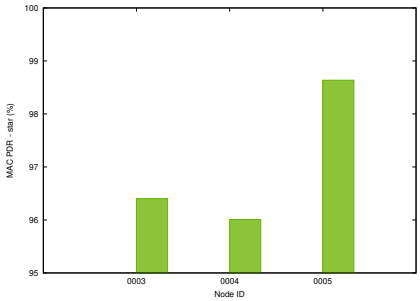


Figure 3.52: Average MAC layer PDR, PGO at IMT Mines Alès, driving.

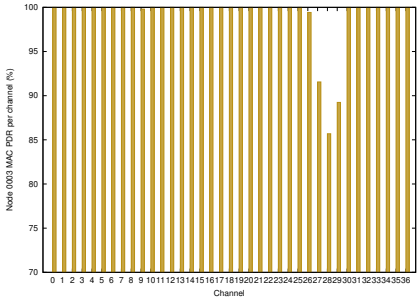


Figure 3.53: Node 3 MAC layer PDR, PGO at IMT Mines Alès, parked.

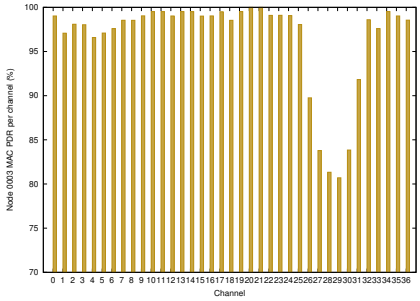


Figure 3.54: Node 3 MAC layer PDR, PGO at IMT Mines Alès, driving.

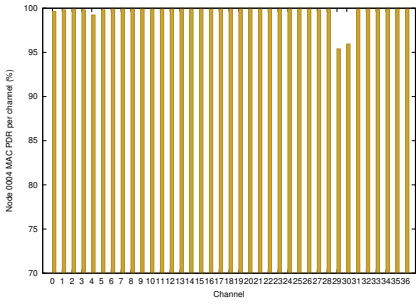


Figure 3.55: Node 4 MAC layer PDR, PGO at IMT Mines Alès, parked.

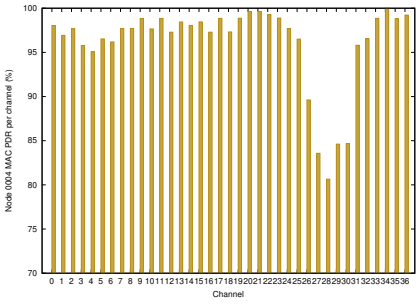


Figure 3.56: Node 4 MAC layer PDR, PGO at IMT Mines Alès, driving.

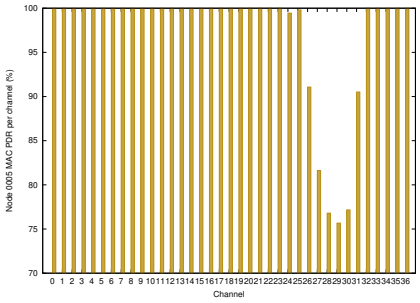


Figure 3.57: Node 5 MAC layer PDR, PGO at IMT Mines Alès, parked.

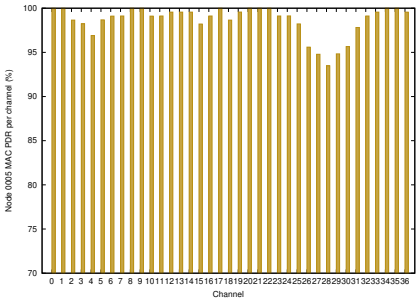


Figure 3.58: Node 5 MAC layer PDR, PGO at IMT Mines Alès, driving.

3.5 CONCLUSION

After the tests described above, we can state that the MAC layer PDR in an EV environment is high (98.5% and above, for all the tests in which we did not generate interfering traffic on purpose). The position of the nodes have an influence on link quality: it seems higher when the nodes are close and in sight of each other.

What seems to cause the more packet losses is the interference with Wi-Fi communication. The IEEE 802.15.4 or BLE channels that overlap with Wi-Fi channels in use have significantly worse PDR than the others. These "bad channels" may evolve when the car is moving, as the neighboring Wi-Fi networks will change. A way to deal with this issue is to use channel blacklisting, as mentioned in Section 6.1.

Driving the car does not seem to have a significant impact on the network. This was very clear on the OSV platform. It was less obvious on the PGO platform, as when driving the nodes suffered some losses that seemed distributed evenly on all channels. But as said, the nodes were not placed where they should have, and this result has little value. Thus, the electromagnetic emissions from the engine should not be a problem for the wireless communications. This is also what can be found in the literature, as discussed in Section 2.4.2.

Figure 3.59: The PGO e-Hemera being driven around the IMT Mines Alès campus during the test.

In the next chapter, we are going to reuse this information to manage the topology of a BMS with wireless communication network, and its TSCH slotframe. As some direct links to the root may be of lower quality than a 2 hops path, we are going to focus mostly on having the best possible topology.

4

NETWORK MANAGEMENT ALGORITHMS

4.1 INTRODUCTION

In Chapter 3, we evaluated the link quality of a WSN in an EV environment. Some links are significantly better than others, and this depends on the position of the nodes. Based on these results, and what is discussed in Section 2.4.3, we will now construct a two level tree topology, trying to use the best possible links available. The application layer packets will be aggregated at intermediate, or rank 1 nodes, in this topology, so the slotframe usage will not increase with this technique, compared to a star topology. It may even be reduced, as less timeslots for retransmissions will need to be allocated.

To achieve this, the first technique we propose is based on LP [73]. It is a well known way of finding an optimal solution to a problem, by maximizing (or minimizing) an OF, and under some constraints. The second technique is based on Simple Descent (SD), a well known optimization technique used in many research fields [72]. In our case, it serves the same purpose, and uses the same OF as our strategy based on LP. It is however more efficient in terms of processing time.

There are many parameters involved in the modeling of the problem. In order to select them properly, we made a software program which implements these techniques, and had it compute solutions for a large set of randomly generated link quality data. With this, it is possible to see what is the effect of a given parameter on the output topology and slotframe, and to select the best value for it.

Contribution

This chapter presents the following contributions:

1. We propose a centralized network management technique based on LP.
2. We propose an improved version of this technique by using the SD technique.
3. We compare both strategies and explain how to choose the parameters used with them.

4.2 MANAGING A BMS WIRELESS NETWORK: WITH LINEAR PROGRAMMING

4.2.1 *Overview*

In a BMS application, the number of nodes is relatively small: it could be as high as 96 if there was one CSU per cell, but 16 network nodes is probably a more realistic number (which is 6 cells per CSU). Also, the number of nodes is known in advance, and all nodes see each other. For these reasons, a centralized solution is chosen for both topology management and scheduling.

We propose a two-steps process to build a topology that uses the best possible links, and then a TSCH schedule. Both steps of the decision process use LP. The goal is to allocate network resources (links and TSCH dedicated time slots) to provide reliability for data delivery of all nodes before the end of a slotframe. The purpose of the presented algorithm is to allocate uplink slots to support the voltage and temperature data collection, which has important timing constraints as mentioned above. In a real deployment, extra timeslots would have to be added to the schedule for other traffic flows that are less constrained. In the two LPs, the variables are integers with binary values. In this scenario, each node must reliably send one data packet to the root per slotframe.

To solve this topology management and scheduling problem, we implemented an application in C++ that takes all the possible links between the nodes with their PDR as input, and print the resulting schedule and topology as output (both graphically and in text form). The PDR for the possible links between the nodes is randomly generated. The graphs shown in Figure 4.1 to 4.3 have been generated by this application. This piece of software relies on the Gnu Linear Programming Kit (GLPK) library to solve the LPs.

In the rest of this section, we describe how we modeled the problem, and to illustrate it, we use a network with 4 CSU nodes plus a root node. However, the method can be applied for any realistic number of nodes for a BMS application.

4.2.2 *Building the Topology*

PROBLEM DESCRIPTION As described above, we create an example initial situation with nodes and possible links between them, as can be seen in Figure 4.1. We then assign a random probability of successful transmission to each link. The random values have been chosen in the $[0.95; 1]$ interval, to resemble the values that have been measured and presented in Chapter 3. We assume the links are symmetrical in terms of quality.

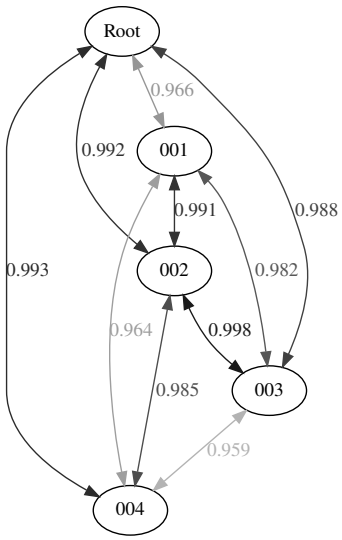


Figure 4.1: Example problem graphical description with 4 nodes.

The variables of the problem are designated with two indexes. The first one is the ID of the transmitter, or source node, and the second one is the ID of the receiver, or destination node for that link. We label N the number of non-root nodes, which gives a total number of nodes, counting also the root, of $N + 1$. So the variables can be represented as in (4.1).

$$X_{i,j} \; i \in [1; N], j \in [0; N] \tag{4.1}$$

In this equation, if the link is used, with i having j as parent, the variable is equal to 1, and it is equal to 0 otherwise.

CONSTRAINTS AND OBJECTIVE FUNCTION In this problem, we try to maximize the average path PDR for each node. We chose to use a coefficient K in the OF we propose to introduce a bias towards using intermediate node (and 2 hops paths), which may help reduce the slotframe length.

In the following, we use the δ variable as follows:

$$\delta_{i,j} = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{otherwise} \end{cases} \tag{4.2}$$

The OF to be maximized is presented in (4.3).

$$F = \sum_{i=1}^{N} \sum_{j=0}^{N} P(i,j) X_{i,j} \tag{4.3}$$

with:

$$P(i,j) = \begin{cases} {p_{i,0}}^2 & \text{if } j = 0 \\ K \; p_{i,j} \; p_{j,0} & \text{otherwise} \end{cases}$$
$$K = 1.1$$

We maximize this OF under the following constraints:

- Each non-root node has a maximum of A children (4.4).

$$\forall i \in [1; N], \sum_{j=1}^{N} \delta_{i,j} X_{i,j} \leqslant A \tag{4.4}$$

- Each node has one parent (4.5).

$$\forall i \in [1; N], \sum_{j=0}^{N} \delta_{i,j} X_{i,j} = 1 \tag{4.5}$$

- Each node has a path to the root (4.6).

$$\forall i \in [1; N],\ \forall j \in [1; N],\ i \neq j,$$
$$X_{i,j} + \sum_{l=1}^{N} \delta_{j,l} X_{j,l} \leqslant 1 \tag{4.6}$$

We have to specify a maximum number of children for the rank 1 nodes because we use aggregation. For this technique to be efficient, we need to be able to transmit the data of a rank one node together with the data of all of its children within one timeslot. If the timeslots are configured to last for 10ms, which is the default, the available payload for application data is 70 bytes in each frame, if encryption is not used. The data contained in an application layer packet from a CSU depends on the number of cells which are monitored by this CSU, and will usually be around 20 bytes. This means, if the available payload size is 70 bytes, and each data packet is 20 bytes, an aggregated packet can contain at maximum the information generated by 3 nodes. Thus, in this scenario, a rank 1 node can have a maximum of 2 children. Which explains why we defined the constant A that depends on the timeslot duration, the use of encryption at MAC layer, and the size of the data packet emitted by the CSUs.

EXAMPLE OUTPUT The program described above ran with the data shown in Figure 4.1 as input gives the result shown in Figure 4.2 as output.

4.2.3 *Building a TSCH Schedule*

PROBLEM DESCRIPTION At this stage of the network management process, the software takes the result that was obtained by the feature described in Section 4.2.2 as input, and outputs the resulting TSCH schedule. It is assumed that the application data payload is small enough (typically 2 bytes per cell voltage measurement and 2 bytes per temperature sensor) to be aggregated at the intermediate nodes,

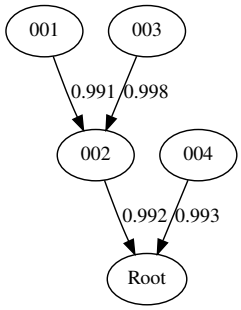


Figure 4.2: Output topology for the 4 nodes example.

and the resulting packet is small enough to be sent within one timeslot to the root node.

The variables are again designated with two indexes. The first one is the ID of the transmitter node, and the second one is the timeslot offset in the slotframe, starting at 0. We do not need to specify the receiver node here, since it has been determined in the previous step. We label S the last timeslot index that may be used. Thus, the variables can be represented as in (4.7).

$$X_{i,k} \; i \in [1; N], k \in [0; S] \tag{4.7}$$

In the following, we label R_1 the set of node IDs that have a rank of 1 in the topology, *i.e.* which have the Root node as parent. For example, nodes 2 and 4 would be of rank 1 in Figure 4.2. We label R_2 the set of node IDs that have a node in R_1 as parent. We label $C_i, i \in [1; N]$ the children nodes of i, that are of rank 2. The generated slotframe always have a first timeslot which is marked as Reserved and which should be used for beacon frames and control plane traffic.

CONSTRAINTS AND OBJECTIVE FUNCTION In this problem, we try to minimize the TSCH slotframe size, while keeping the path PDR of every node above a given threshold. This OF is described by (4.8).

$$F = \sum_{i=1}^{N} \sum_{k=0}^{S} (k+1) X_{i,k} \tag{4.8}$$

In this OF, we make the "cost" of scheduling timeslots increase linearly over time. What matters is that this cost increases, not how (as this "cost" does not have a real meaning), to give incentive to the system to schedule the timeslots as early as possible to reduce the portion of the slotframe used for the application traffic.

We try to minimize this OF under the following constraints:

- If a node is sending its data packet through a relay node to the root, it must send it to the intermediate node before that node sends the aggregated data packet to the root. In other words,

we impose a happens-before relationship between receiving and sending for the R_1 nodes (4.9).

$$\forall i \in R_1,\ \forall j \in C_i, \forall k_1 \in [0;S], \forall k_2 \in [k_1;S], \\ X_{i,k_1} + X_{j,k_2} \leqslant 1 \tag{4.9}$$

- The channel offset is limited to C channels (4.10).

$$\forall k \in [0;S], \sum_{i=1}^{N} X_{i,k} \leqslant C \tag{4.10}$$

- Each node can only use its radio for one dedicated timeslot for a same slot offset. It is sufficient to input this constraint for nodes in R_1 (4.11) and for the root (4.12).

$$\forall i \in R_1,\ \forall k \in [0;S],\ X_{i,k} + \sum_{j \in C_i} X_{j,k} \leqslant 1 \tag{4.11}$$

$$\forall k \in [0;S],\ \sum_{i \in R_1} X_{i,k} \leqslant 1 \tag{4.12}$$

- Each node must have at least one transmit timeslot scheduled (4.13).

$$\forall i \in [1;N] \sum_{i=1}^{N} X_{i,k} \geqslant 1 \tag{4.13}$$

- The path to the Root for each node must have a PDR above the given threshold (4.14).

$$\forall i \in [1;N] \sum_{k=0}^{S} X_{i,k} \geqslant T_i \tag{4.14}$$

where T_i is the minimum number of transmissions for the link with sender node i to meet the minimum path PDR threshold requirement.

In equation (4.14), to calculate the number of transmission timeslots to be allocated for each link, we consider the PDR without retransmission of each link, and the minimum path PDR. To do this, we iteratively schedule a retransmission on the worst link of the worst path, until all the path PDR are above the requested threshold. After the retransmissions have been scheduled, the resulting link PDR is as shown in (4.15).

$$PDR_{rtr} = 1 - (1 - PDR)^{tr} \tag{4.15}$$

In this equation, PDR_{rtr} is the PDR after all the transmissions have been scheduled, and tr is the total number of transmissions for that link.

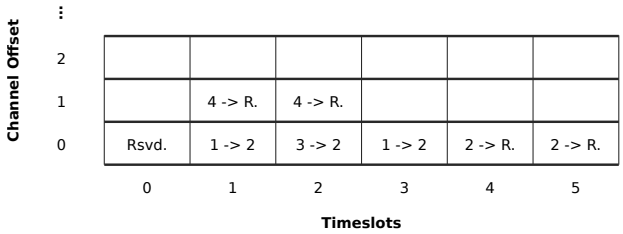


Figure 4.3: Output schedule for the 4 nodes example.

EXAMPLE OUTPUT The program described above ran with the data shown in Figure 4.2 as input gives the result shown in Figure 4.3 as output. In this example, the minimum allowed path PDR has been arbitrarily set to 99.5%. Thanks to the retransmissions, the path PDR to the root for node 1 to 4 is 99.985%, 99.994%, 99.794% and 99.995% respectively.

4.2.4 *Evaluation of the Linear Programming Solution*

As latency is a major concern in a BMS application, the output slotframe size is a relevant metric to see how this proposed solution behaves compared to simpler strategies. To that extent, we use our application to build schedules for a network of between 2 and 16 nodes, with random link quality in the 95 to 100% range. For each number of nodes, the test is run 10 times, with new link conditions every time. In this test case, we manage the network with 3 strategies. The first one is Star topology. The second one is a strategy using only aggregation at intermediate (or rank 1) nodes, and which minimizes the slotframe size. The last one is our proposed solution. While the first two strategies do not consider the link PDR and do not schedule any retransmission, ours is set to build a minimum path PDR of 99.5%. In this test, the rank 1 nodes, those which have a direct link to the root, can have a maximum of two children.

The results, displayed in Figure 4.4, show that even though for very small networks our solution uses more timeslots, when the network grows it requires a smaller portion of the slotframe than Star topology, although it has to guarantee a maximum path Packet Error Rate (PER) 10 times smaller than the maximum link PER, with the settings used. Moreover, our solution gives slotframe sizes slightly above the strategy that only tries to minimize them. This shows that wisely choosing the links and using retransmissions when needed can significantly improve performance at a reasonable cost.

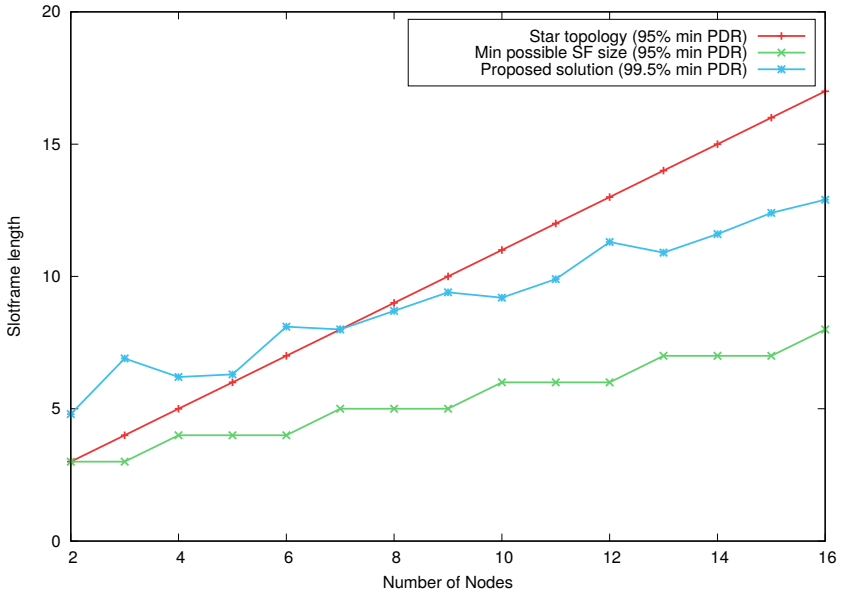


Figure 4.4: Comparison of average slotframe length between Star topology, Maximum Aggregation topology, and the proposed solution.

4.3 MANAGING A BMS WIRELESS NETWORK: WITH SIMPLE DESCENT

4.3.1 *Problem description*

The strategy exposed above gives an optimal result under the selected constraints, but building the topology may take time in some situations, which is a well-known downside of LP. In this section, we use the same modeling for the problem, but use the SD method to build the topology. We only focus on building the topology, because an optimal schedule can be built in a procedural way, by allocating all the slots of the nodes with rank 2 to the most left available slots, then allocating the slots for all the nodes of rank 1 to the most left available slots, while taking into account the constraints exposed in Section 4.2.3.

This is viable because the OF for this problem will be at a minimum when the slots are allocated at the most left side of the schedule possible. Scheduling the rank 2 nodes before the rank 1 nodes makes the solution compliant with the first constraint, describing a happen-before relationship. From there, it is possible to build an algorithm that tries to schedule a slot for a node in the first slot, check all the constraints, and move to the next slot to its right, if not all the constraints were satisfied, and repeat this until it finds a slot that fulfills all the constraints.

4.3.2 *Proposed solution*

In the simple descent method, the goal is to rapidly find an approximate solution to the problem. This is done by first building a valid solution, and then applying permutations to it to keep this intermediate result a valid solution at all iterations, while converging towards a local minimum or maximum, according to the objective function.

The technique described below has been implemented into the same application presented in Section 4.2.

To build an initial solution, we used four methods, described below. In the associated examples, the green color means it has been selected, and the red color means the link has been removed from the available links list. The numbers reflect the order in which these operations are performed.

- **Star topology**. All the nodes have a link to the root. An example is available in Figure 4.5.
- **Best link first**. This is an iterative process in which the best link is selected. If it is a link to another non-root node, this other node has its link to the root selected, and the other possible links to the now rank 2 node are removed from the available link list. An example of the first iteration of this algorithm is available in Figure 4.6.
- **Best link to root first**. This is an iterative process in which the best link to the root are selected first, until the root has the minimum possible child nodes. The remaining links to the root are removed from the link list. Then, the remaining nodes, which are then of rank 2, have their link selected using the "best link first" method described above. An example of the first iterations of this algorithm is available in Figure 4.7.
- **Eliminate weakest link first**. This is an iterative process in which the worst links are eliminated. Every time a link is removed from the link list, the algorithm checks if for each node they have more than one path to the root. When a node has only one possible path left, these links are selected. An example of the first iterations of this algorithm is available in Figure 4.8.

In the second part of this process, we apply permutations to the current solution and calculate the value of the objective function defined in equation 4.3 for each permutation that has been tried. The permutation with the highest objective function value becomes the current solution for the next iteration. When no permutation gives a better result than the current solution, a local maximum has been reached and the current solution then becomes the topology.

The following permutations have been used to try to approach the optimal solution:

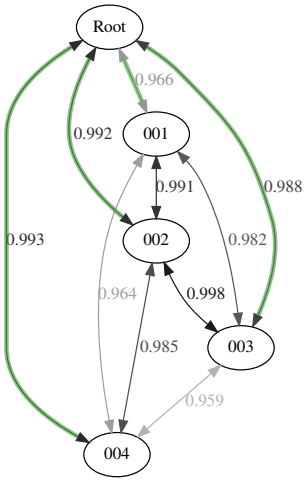


Figure 4.5: Star topology with 4 nodes example.

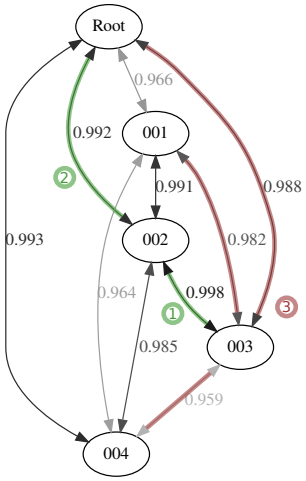


Figure 4.6: First iteration of the "best link first" initial solution method, with 4 nodes example.

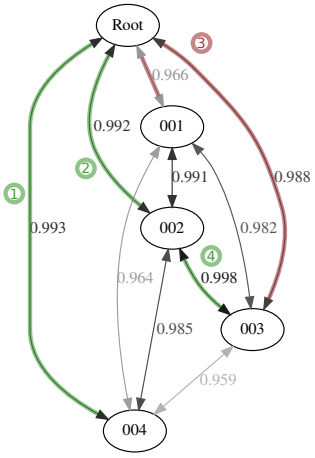


Figure 4.7: First few steps of the "best link to root first" initial solution method, with 4 nodes example.

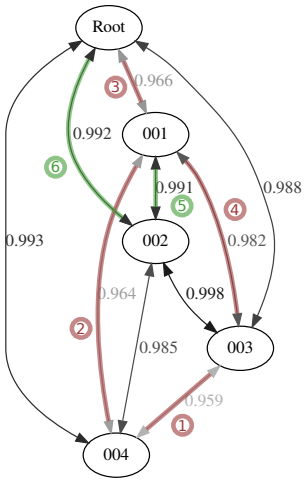


Figure 4.8: First iterations of the "eliminate weakest link first" initial solution method, with 4 nodes example.

- **Exchange a rank 1 node with a rank 2 node**: switch the position of a rank 1 node with a rank 2 node in the topology.
- **Exchange two rank 2 nodes**: switch the position of two rank 2 nodes in the topology.
- **Make a rank 1 node a child of another rank 1 node**: select a node with rank 1 with no child, and make it rank 2 as child of another rank 1 node.

It is important to note that the resulting topology obtained with this method may vary depending on the initial solution chosen and the set of permutations used.

4.3.3 *Evaluating the result*

Several ways to evaluate the obtained solutions can be used, like average path PDR or minimum path PDR. But as aggregation of data packets may happen at rank 1 nodes, the most meaningful is probably the average weighted link PDR, as in this case the PDR for a link will be counted as many times as there will be data packets in the aggregated packet. For example, if a rank 1 node has 2 children, its link to the root PDR is counted 3 times, to reflect the importance of the packet. This is the method we used to evaluate a solution *a posteriori* in the rest of this chapter. It would have actually been interesting to use this function as the OF in the LP technique, but that did not seem feasible without making the problem non-linear, which we wanted to avoid.

4.3.4 *Choosing the initial solution*

To get the best chances at finding a good solution, according to the average weighted link PDR metric, we compared how the different initial solutions performed. The four initial solution building methods have been tested against situations with random link qualities in the $[0.95; 1]$ interval, for a number of nodes varying from 2 to 32, and 100 times for each number of nodes. The results are displayed in Figure 4.9.

The best results are obtained with the star topology as the initial solution when the number of nodes is low, and then the "best link to root" method gives results that are similarly as good when the number of nodes increases. In the remainder of this section, we use the star topology to build the initial solution.

The link quality goes higher when the number of nodes increases on this plot, because the more nodes, the more the system has opportunities to select the better links.

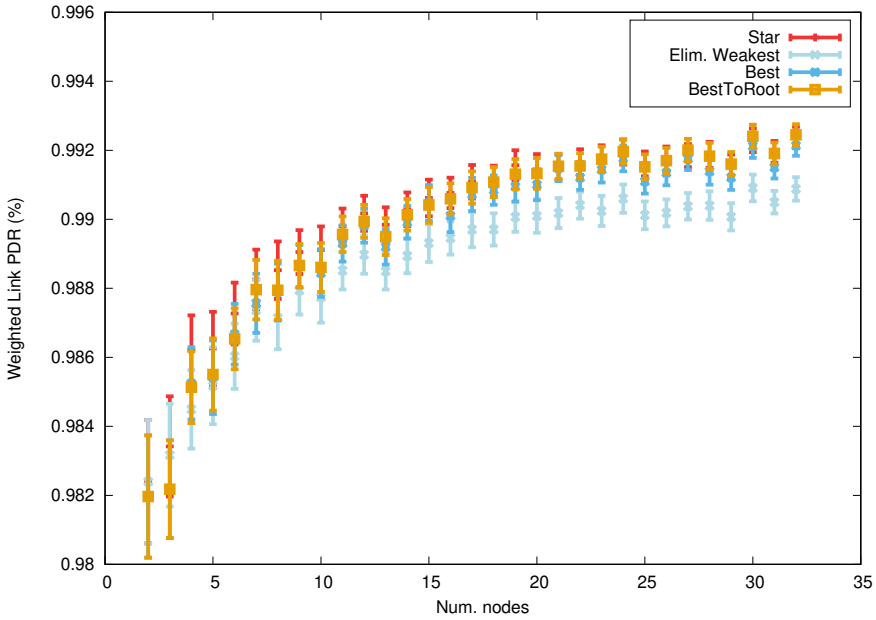


Figure 4.9: Comparison of the weighted link PDR for the four initial solutions that have been tried, for a number of nodes from 2 to 32, with a 100 initial link situations in every case.

4.4 PERFORMANCE EVALUATION AND PARAMETERS TUNING

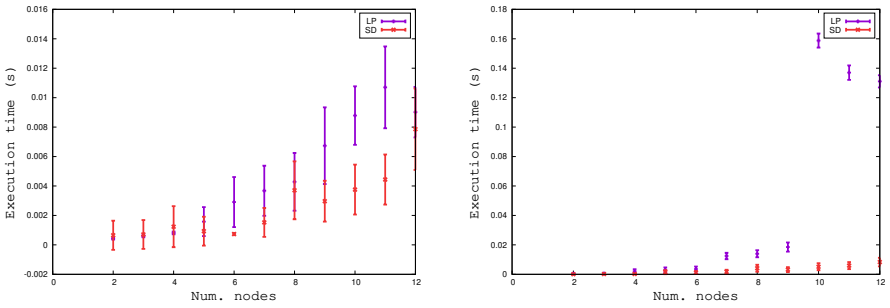
In order to know if the SD technique is viable, we compared it to the LP technique in several situations. The metrics that have been used are the weighted link PDR, and execution time required to build the solution.

First we keep the constant K to 1.1 and the objective function as in equation 4.3, to keep a bias towards using better links for those which involve the root node, as the MAC layer frames going through these links will usually contain more than one data packet. The value to use for these coefficients is actually investigated and discussed below. For each number of nodes, the two strategies are tested against 2000 initial situations. Here we take a look at the execution time for both strategies. The link PDR is chosen randomly in the $[0.7; 1]$ interval in Figure 4.10a, and in $[0.95; 1]$ interval in Figure 4.10b.

In both figures we can see that the execution time becomes much longer for the LP method when the number of nodes increases. This difference would become even more meaningful on an embedded CPU with constrained resources.

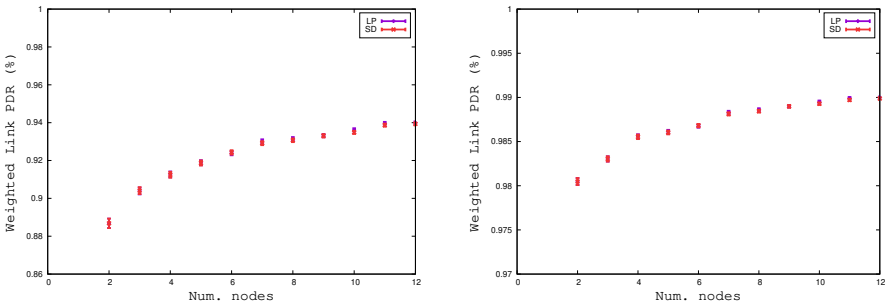
During this test, the metrics to evaluate the different solutions were also calculated. In Figure 4.11, we compare the average weighted link PDR for both techniques.

These figures show that the LP techniques gives slightly better results. However the difference does not seem to be significant.



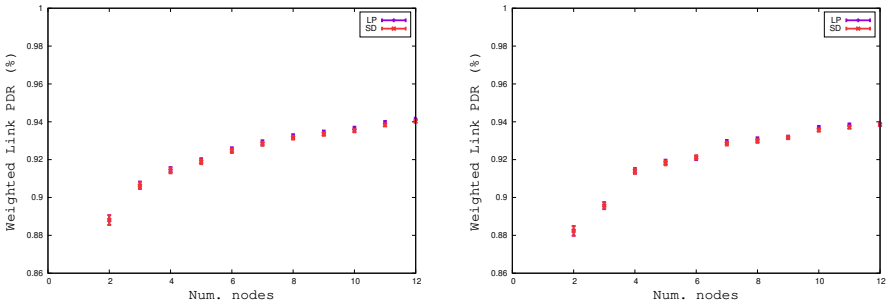
(a) Link PDR in $[0.7; 1]$. (b) Link PDR in $[0.95; 1]$.

Figure 4.10: Comparison of topology building time between LP and SD, for 2000 initial situations.



(a) Link PDR in $[0.7; 1]$. (b) Link PDR in $[0.95; 1]$.

Figure 4.11: Comparison of topology average weighted link PDR for LP and SD, for 2000 initial situations, and $K = 1.1$.



(a) With K = 1.0.

(b) With K = 1.5.

Figure 4.12: Comparison of topology average weighted link PDR for LP and SD, with link PDR in [0.7; 1] for 2000 initial situations.

Similar tests have been ran with a link PDR in [0.7; 1], and K equals to 1 and 1.5, to see how much this parameter impacts the result. The results can be seen in Figure 4.12. They show that there is no significant upside in using this K parameter in the first place, or that the best value for it is 1.

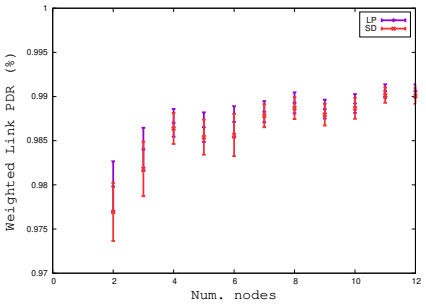
Finally, other tests have been run, in which the exponent of $p_{i,0}$ in equation 4.3, which is the probability of a successful transmission from node i to the root, has been changed, to see if this was efficient. This exponent is set to 1 in Figure 4.13a, to 2 in Figure 4.13b and to 3 in Figure 4.13c. It seems that at least for the SD technique, using an exponent of 2 gives better results than 1, and there is no significant difference between 2 and 3.

From all these results, we can conclude that even though both techniques are efficient, the SD technique with K = 1.0, and an exponent of 2 for $p_{i,0}$ in equation 4.3, is nearly as good as LP and requires a lower computation time. Thus, SD is the topology management technique we will use in the rest of this work.

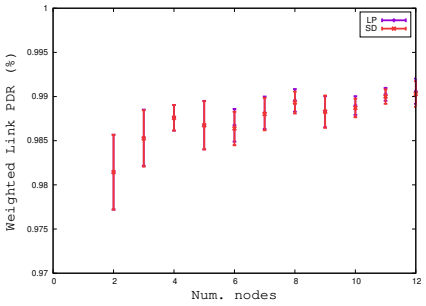
4.5 CONCLUSION

In this chapter, we proposed two strategies for network management. The first method is based on LP, and while it does generate a high quality topology, it can be costly in terms of processing time in some situations, especially when all the links are of high quality with a PDR close to 1. The second method, based on SD, gives results that are almost as good with a much lower computation time, and could even be implemented to be limited to a bounded number of operations if that is necessary.

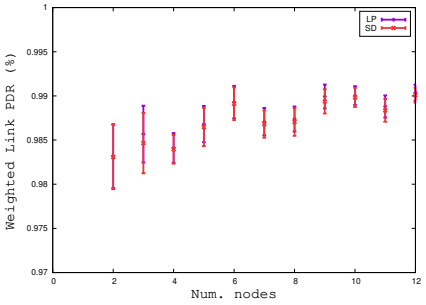
However, the techniques proposed here can not be used as is to manage a network, as they assume that the network manager has perfect information on all the possible links quality. A mechanism to give the PDR data to this central controller, and to propagate its decisions is required. Also, when the network is started for the first



(a) Exponent of $p_{i,0}$ in equation 4.3 is 1.



(b) Exponent of $p_{i,0}$ in equation 4.3 is 2.



(c) Exponent of $p_{i,0}$ in equation 4.3 is 3.

Figure 4.13: Comparison of topology average weighted link PDR for LP and SD, with link PDR in $[0.95; 1]$ for 2000 initial situations, $K = 1.0$.

time, the possible links quality are unknown. Thus, an iterative process to try the links and discover their PDR should be used. This is what is discussed in Chapter 5.

5

WIRELESS NETWORK FOR BATTERY MANAGEMENT PROTOCOL

5.1 INTRODUCTION

In order to propose a solution that would be suitable for a real BMS with wireless communications, we introduce in this chapter a new routing and scheduling management protocol that we designed and developed. It also allowed us to test the algorithms presented in Chapter 4. We describe the hardware and software architecture we used to implement and test this protocol and algorithms, the network stack, the joining sequence for a participating node, and the control messages exchanged. The algorithms presented in Chapter 4 assume a perfect knowledge of the links quality. Here these algorithms are used in an iterative way to explore the links quality through various topologies, and converge towards the best possible one. Moreover, we present the experimental test results that we obtained inside a battery pack with this setup. These results show that, by actually selecting the best available links to build a multi-hop network, and providing enough retransmission timeslots, a very high QoS can be achieved. Finally, we discuss the upsides, and possible improvements for this protocol.

Contribution

This chapter presents the following contributions:

1. We propose an architecture for centralized network management with the Contiki-NG OS.
2. We propose and describe a new protocol for control plane, network discovery and management, to be used in Contiki-NG in place of RPL to propagate the decisions of the centralized network manager.
3. We present the performance evaluation results of the experiment that we performed with this protocol with a network of I3Mote nodes inside a battery pack.

5.2 INCENTIVES FOR CENTRALIZED MANAGEMENT AND INTENTION

As is presented in Chapter 4, centralized management has been chosen, to make decisions based on full information about the network link qualities. Using RPL to propagate the decisions of the centralized network manager was not an option, as RPL has been designed for distributed and local decision making, based on an OF run on the network nodes directly. So the messages exchanged in RPL do not match what are trying to achieve here.

In the solution we propose, we have chosen to divide roles between the root node and the network manager. The root node is in charge of managing the nodes state, especially during the joining procedure, whereas the network manager only considers nodes that have joined and are ready, and processes their link quality data to make topology and schedule updates decisions.

In RPL, all control messages can be routed, and Destination Information Objects (DIOs) in particular. In our situation, we based our work on the assumption that all nodes are one hop away from each other, and will have reasonable link quality with any other node (that is, above 90% PDR). This is, at least, what has been observed in the experiments presented in Chapter 3. So the control message that is equivalent to DIOs in the protocol we propose, which we name BMS Network Advertisement (BNA), is sent only from the root, on the multicast IPv6 address used for that purpose. It will result in a broadcast message at the MAC layer.

This makes the protocol simpler, both in design and in how it operates. This protocol, although it is aiming to achieve close to deterministic communications for application layer packets, is best-effort in terms of network management. It takes advantage of the fact that any kind of communication can be performed within one hop, and if a control packet is missed, any receiver node in this situation will be able to re-synchronize with the next control message of the same kind it receives, as almost all control messages are sent periodically. In other words, the protocol has been thought to work based on asynchronous events and be reactive. The other possible approach would have been to use acknowledgments for control packets (such as DIO Acknowledgements (ACKs)), but this is not necessary here, and would have made the protocol heavier.

5.3 ARCHITECTURE

5.3.1 *Network and software stack*

The protocol we propose has been designed to be used in place of RPL, to set how routing should be performed in the nodes, according to the

Protocols
Custom protocol messages
ICMPv6
6LowPAN
IEEE 802.15.4-2015 TSCH (MAC layer)
IEEE 802.15.4-2015 TSCH (PHY layer)

Table 5.1: Custom routing protocol control messages stack.

Protocols
BMS application messages
UDP
6LowPAN
IEEE 802.15.4-2015 TSCH (MAC layer)
IEEE 802.15.4-2015 TSCH (PHY layer)

Table 5.2: BMS protocol application messages stack.

Software layer	Driver
`NETSTACK_ROUTING`	`bmsrouting`
`NETSTACK_NETWORK`	uIP / 6LowPAN driver
`NETSTACK_MAC`	TSCH driver
`NETSTACK_RADIO`	Set by platform

Table 5.3: Software layers for network operation that we use in Contiki-NG.

network manager's decisions. We took advantage of how the Contiki-NG code separates the implementation of the different software layers it is made of, and makes the drivers generic, to define a new routing driver. This is what is shown in Table 5.3. The custom software layer we use in place of RPL is named `bmsrouting`. The protocol stack used for control messages is described in Table 5.1. The protocol used to carry these messages, Internet Control Message Protocol for IPv6 (ICMPv6), has been chosen to be as close as possible to what RPL does. Like in RPL, some messages are multicast, and others are unicast.

The application layer messages use the typical IoT stack, as shown in Table 5.2. Although CoAP could have been used, none of its features are really helpful in the BMS scenario, and it would have brought extra overhead. Thus a custom binary message format over UDP has been chosen.

The TSCH mode of IEEE 802.15.4 is used in what we developed, although the routing part of the protocol would be suitable for CSMA as well. This means a scheduling mechanism is used. All nodes, including the root, start with a slotframe of 25 timeslots, with a default length of 10ms each, and one shared timeslot at slot offset 0, channel offset 0. When the network manager makes topology update decisions, it will also schedule at least one uplink timeslot from each node to its parent, according to the algorithms presented in Chapter 4. These dedicated transmit timeslot to the parent, or receive timeslot from the child are configured upon reception of a BMS Network Topology

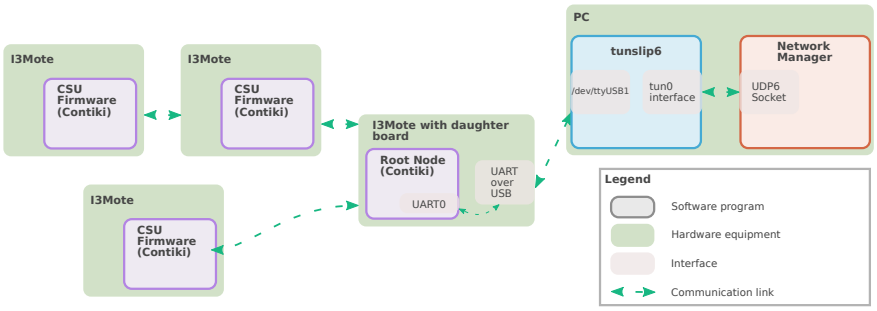


Figure 5.1: BMS with wireless communication network and software architecture used during the final test, with I3Mote nodes.

Update (BNTU) message for non-root nodes, and a Network Manager Update (NMU) message for the root node.

5.3.2 *Target platform*

The purpose of the protocol and system that is described in this chapter is to run an experiment with real hardware, and with a setup that should be similar to what we would use in a real BMS with wireless communications implementation. To be able to compare the results obtained here with the tests that have been performed in Chapter 3, the target hardware platform for the IoT nodes is CC2650 I3Motes. Although in a real deployment the MCU running the network manager and MCU software would be an embedded device, here we used a PC for convenience. The root node runs a Contiki-NG project we made, called `bms-br`, which is derived from the `rpl-border-router` project. In the end, our setup is very close to what is used in a typical Contiki-NG and RPL network, with the `tunslip6` application being the bridge between the IoT world and the IPv6 world. This setup is displayed in Figure 5.1.

What is particular here, and different from a real BMS with wireless communications implementation, is that there is no dedicated MCU program. The application layer messages from the nodes, which do not contain actual data about a battery pack, are sent to the network manager application, in order to monitor the application layer packet losses. This is because we want to monitor the network performance, and evaluate the network management algorithms quality, and do not need to supervise an actual battery pack.

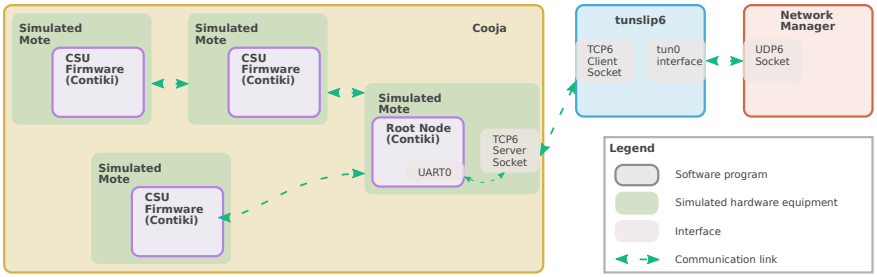


Figure 5.2: BMS with wireless communications network and software architecture used during development, with Cooja.

5.3.3 *Platform used during development: Cooja*

In order to make the setup described above possible, we developed and tested our network protocol in the Cooja simulator. As of today, the only hardware targets that are available in it are rather old and with limited memory, so we used the Cooja Mote target. The setup that has been used is very close to the `rpl-border-router` example, except we used our own Contiki-NG projects, and the `bmsrouting` routing layer to manage the network, as shown in Figure 5.2.

5.4 MESSAGES AND OPERATION PRINCIPLE DESCRIPTION

5.4.1 *IoT network control messages*

The following messages are sent over the IEEE 802.15.4 TSCH network, as ICMPv6 messages, either on a predefined multicast address, or a node's unicast IPv6 address. They are not routed over the USB link towards the network manager. They are sent by our custom routing layer on reception of another routing layer message, or when a timer expires.

5.4.1.1 *BMS Network Advertisement*

The role of this message is to advertise the network, and help the nodes that participate to remain synchronized. It is sent by the root node every two seconds, with a random delay added, on the multicast IPv6 address used by our routing layer. This message serves the same purpose as a DIO packet in RPL, the main differences being that it can only be sent by the root node, and that the delay between the packets sent does not change over time (whereas RPL uses a trickle timer). BNA messages are periodically sent by the root node, every 3 seconds.

This message contains the following information (in order):

- **Root IP** (16 bytes): The global IPv6 address of the root node.
- **Topology counter** (1 byte): A counter to help a node see if it has missed a BNTU message.
- **Network prefix information** (22 bytes): Network prefix, with its size and lifetime, for the node to create its own global IPv6 address within the current network.
- **BMS data flag** (1 byte): When this flag is set, the nodes are allowed, and should, send BMS application layer packets containing their cell voltage and temperature data to their parent. Actual data packet format in the current implementation are described in Section 5.4.3

5.4.1.2 *BMS Network Join Request*

This message is sent by a non-root node, upon reception of a BNA message, when it is trying to join the network.

This message contains the following information:

- **Node IP** (16 bytes): The global IPv6 address of the node which is trying to join, configured using the prefix received in a BNA.

5.4.1.3 *BMS Network Topology Update*

This message is sent by the root node, upon reception of a BMS Network Join Request (BNJR) message, or after a NMU is received, because the network manager has made an update decision. The main purpose of this message is to tell a node what parent it should have, but also what should be its dedicated send and receive timeslots.

Sending this message can also be triggered by the reception of a BMS Network Topology Information (BNTI) by the root node, in which the parent ID information would not match the supposed parent ID of the node (meaning the node has missed a BNTU).

This message contains the following information (in order):

- **Node own IPv6 link local address** (16 bytes)
- **Node parent IPv6 link local address** (16 bytes)
- **Node ID** (1 byte)
- **Parent ID** (1 byte)
- **Rank** (1 byte): this value is either 1 if parent ID is 0 (meaning the parent is the root), and 2 otherwise. This is actually redundant with the Parent ID field.
- **List of transmit slots to parent**, as follows:
  - **Number of items in the list** (1 byte)

  - **Items** (2 bytes each): first byte is the timeslot offset in the slotframe, second byte is the channel offset.

- **List of receive slots from children**, as follows:
  - **Number of items in the list** (1 byte)
  - **Items** (2 bytes each): first byte is the timeslot offset in the slotframe, second byte is the channel offset.

Please note that in the second list, the ID or IPv6 address of the child node is not specified. This is not required to configure a receive timeslot. If a frame is received during such a timeslot, the MAC layer will either discard it if the unicast destination address of the frame is not this node, or forward it to the upper layer.

5.4.1.4 *BMS Network Topology Information*

This message is periodically sent by each non-root node, every 10 seconds (with a random delay of up to 1 second added to limit the risk of contention), on the multicast address used by our BMS routing protocol management. Its goal is mainly to inform the root node what is its current parent (to help the root node see if this particular node has information about the topology that is synchronized with its own information), but also to tell the root node about the link quality the node has with its parent.

This message contains the following information (in order):

- **Node ID** (1 byte)
- **Parent ID** (1 byte)
- **Rank** (1 byte): this information is redundant with parent ID.
- **Topology ID** (1 byte): this should match the value sent in the BNA messages. If this ID is lower, a BNTU should be sent to that node.
- **Sequence Number** (1 byte): this is to help the root node see if it has missed one or more BNTI from this node.
- **MAC transmitted packets** (2 bytes): number of transmitted MAC layer packets since the previous BNTI has been sent.
- **MAC lost packets** (2 bytes): number of lost MAC layer packets since the previous BNTI has been sent.
- **MAC transmitted packets – old 1** (2 bytes)
- **MAC lost packets – old 1** (2 bytes)
- **MAC transmitted packets – old 2** (2 bytes)

- **MAC lost packets – old 2** (2 bytes)

The fields that are labeled "old 1" are a duplicate of the same field from the BNTI that was sent just before. Similarly, the fields that are labeled "old 2" are a duplicate of the same fields from the BNTI sent even before. Thus, even if the root node misses a significant amount of these messages, it still has all, or almost all, the data about the node. In other words, each BNTI, which is sent approximately every 10 seconds, contains the MAC layer statistics of that node for the last 30 seconds, approximately.

It would have been possible to do otherwise, using ACK messages from the root node on a successful BNTI reception, but this would have represented more overhead for the network. Especially because the data is so small: 4 bytes generated every 10 seconds.

5.4.2 *Messages between root node and Topology Manager*

The following messages are exchanged between the root node and the network manager over the serial over USB link, and are UDP over IPv6 packets, using unicast addresses. Please note that a specific and fixed port is used on both sides for this datagram socket, which is different from the port used by application layer messages. The condition to be met for sending each of these messages is described below.

5.4.2.1 *Network Manager Information*

This message is sent by the root node to the network manager upon reception of a BNTI, and without delay. In this situation, the root node just acts as a gateway between the node and the network manager. After it has processed the information contained in the BNTI for itself.

This message contains the exact same information as a BNTI, in the exact same order.

5.4.2.2 *Network Manager Update*

This message is sent by the network manager to the root node right after an update decision. Again, it is then the responsibility of the root node to make sure that the other nodes in the network keep up with this decision and remain synchronized. This message contains two lists. The first one is the topology information, and the second one is the schedule information.

This message contains the following information (in order):

- **Nodes list size** (1 byte): number of elements.
- **Nodes list**: 2 bytes per element, each element is represented as:
  - **Nodes ID** (1 byte)

  - **Parent ID** (1 byte)

- **Schedule list size** (2 bytes): number of elements.
- **Schedule list**: 3 bytes per element, each element is represented as:
  - **Nodes ID** (1 byte)
  - **Transmit slot to parent, slot offset** (1 byte)
  - **Transmit slot to parent, channel offset** (1 byte)

Please note that the second two lists combined are enough for the route node to determine which node should configure received slot in its slotframe, and at which location.

After receiving such a message, the root node will update its local representation of what the topology should be, and start a short timer. When this expires, the root node will send a BNTU to another node that needs to change its position in the topology, or slotframe settings. When this message is sent, the root node resets its BNTU timer to prepare to send the next one to another node that needs update. When the BNTU timer expires, the next node that will be updated is preferably a node that should be of rank 1 in the new topology, among all the nodes that need an update.

5.4.3 *Application layer messages*

As the goal here is to make a proof-of-concept, and not actually supervise a battery pack, the application layer messages contain only minimal information. They have actually two purposes in our case: generate traffic from the non-root nodes, to have MAC layer PDR statistics, and count them at the network manager to see what is the application layer PDR for each node. The goal being of course to have 100% application layer PDR once the network is formed and stable.

This message contains the following information (in order):

- **Nodes ID** (1 byte)
- Random data (2 bytes)
- **Sequence number** (1 byte)
- **Child node 1 ID** (1 byte)
- Random data (2 bytes)
- **Sequence number** (1 byte)
- **Child node 2 ID** (1 byte)
- Random data (2 bytes)

- **Sequence number** (1 byte)

If the node has no child, the application layer packet will only contain the first 4 bytes described above. If the node has one child, the application layer packet will only contain the first 8 bytes described above. Else, the node has two children, and its application layer packet contains the full 12 bytes described previously.

If the node has rank 1 (meaning it has the root as parent), it will send its data directly to the network manager, together with the data it has received from its children, if any, in an aggregated packet. Otherwise, it sends its data to its parent. As rank 2 nodes can not have children in our system, their application layer packets are only 4 bytes long.

Once a node receives a BNA with the `BMS data flag` set to 1, it starts sending one application layer packet per slotframe.

In the packet format described here, there are fields with random data. Actually they are bytes with fixed values, above `0xf1`. It happened that sometimes during the tests the `tunslip6` application would crash when receiving packets with series of bytes equal to, or close to, 0. This is actually a workaround, and these bytes are of no interest for the rest of the applications.

5.4.4 *How the packets are actually routed*

The actual routing of the packet is performed by the classical software layer used by Contiki-NG for routing: `uIP`. The mechanisms we use for handling the routes configurations are exactly the same as in the `RPL Lite` implementation. This means, when a BNTU is received by a node, it will set the IPv6 link local address of its parent as its default route. This is enough to handle routing of uplink traffic. Downlink traffic (if any) is handled by the root node, using its current representation of the network, and is performed using source-routing. This is reasonable in our case, since all nodes are two hops away from the root, and the platform we developed is still a testbed that has no use-case for downlink traffic yet.

5.4.5 *Network manager behavior regarding link quality estimation and update decisions*

The network manager has been designed to compute the best possible topology based on the link information it has. However, the link quality information comes to it periodically in samples, and is only about the link that are currently being used. So it is necessary to both have mechanisms that give it incentives to explore all the possible links, but also to mitigate the fact that some link quality data may not be fresh.

5.4.5.1 *When network manager update decision is triggered*

Internally, the network manager has two main functions. One that is called upon reception of a Network Manager Information (NMI) packet, which records the link information of a specific node to its parent. The other, which is called every 20 seconds, computes a new topology based on the current link quality information. This computation is done with the algorithm described in Chapter 4. The result is not only a topology and schedule, but also a value for the objective function.

The value obtained for the objective function is then compared to the values obtained in the last 4 iterations of that function call. If the difference between the current objective function value is bigger than any of the 4 values calculated previously, an update decision is taken.

This is one way to handle the trade-off between updating the topology to discover what the other links are like, and giving the network time to adapt and gather a significant amount of data before making an update decision.

5.4.5.2 *How link quality is calculated*

The link quality may vary over time, mainly due to interference, as exposed in Chapter 3. But having old link quality data is better than no data at all. So we implemented a function that calculates link quality in the network manager by giving more weight to the more recent data.

We compute link quality based on the following assumption. A lost packet is most often due to an acknowledgment that has not been received. With this, it is hard to know in which direction a packet has been lost. So, in our system, the links are supposed to be symmetrical.

The PDR is the ratio of the packets that have been received over the packets that have been transmitted. Here, we compute a weighted average of the PDR data samples that have been received from the node, or the two nodes, that may participate in a specific link.

In the calculation of the PDR, each data sample is attributed a coefficient, which is calculated with equation 5.1. i is the index of the PDR data sample in the list of samples received for this link, Ts is the time of reception of the data sample, and N is the number of data samples received for this link. This coefficient is also bounded to the interval $[0.1; 1]$.

$$C_i = max(\frac{Ts_i - Ts_0}{Ts_N - Ts_0}, 0.1) \tag{5.1}$$

In what follows, the number of transmitted packets for data sample i is referred to as Tx_i, and the number of failed transmissions for that sample is referred to as F_i. This gives the formula presented in equation 5.2 to calculate the weighted PDR for a link.

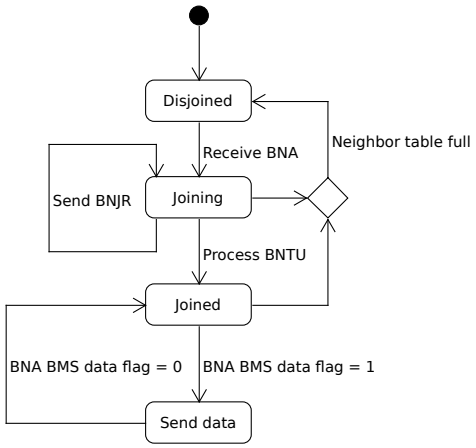


Figure 5.3: Non-root node joining state machine.

$$PDR = 1 - \frac{\sum_{i=1}^{N} C_i \times F_i}{\sum_{i=1}^{N} C_i \times Tx_i} \tag{5.2}$$

With this system, the old data samples have little influence over the final decision, but are still taken into account. The number of samples kept for each link is bounded, and when the list of samples is full and a new sample is received, the oldest sample is discarded, and then the new sample is added to the list. In our implementation, samples can not be discarded based on their timestamps. The only way to discard a sample is by having it "pushed out" of the list by a new one.

5.4.6 *Node joined state*

The possible states a node can go through is pictured in Figure 5.3. On start-up, and after joining the IEEE 802.15.4 network with the configured PAN ID, a node starts listening for BNA messages. When a BNA message is received, the node uses the provided information to configure the root node address, and topology version parameters. This makes it switch from the disjoined to joining state. It also sends a BNJR message. Upon BNTU reception, it finishes to configure its network parameters, mainly parent ID and thus default route, and switches to the joined state. If, at any of the two stages mentioned here, it would have its neighbor cache full, it will clear this cache and switch back to the disjoined state. However, this situation is very unlikely to occur. In the joining state, if no BNTU is received after the expiration of a timer, in our case set to 8 seconds, it will resend a BNJR.

When the node has successfully joined, it continues to listen for BNA messages. If the `BMS data flag` in this message is set to 1, it will start sending application layer messages, at a rate of one per slotframe. Should this flag be set to 0 again, it will stop sending these messages.

5.5 NODE JOIN SEQUENCE

An ideal join sequence for a BMS CSU node is presented in Figure 5.4. In this scenario, when the network starts, the root node starts sending Enhanced Beacon (EB) frames. As soon as all of its network layers are ready, these advertisement packets contain network information as well, like this is done in RPL with DIOs, and the period of these advertisement messages is also set by the network layer. In our case, these messages are BNA, sent in broadcast at the MAC layer, and multicast at the network layer. Meanwhile, the node is scanning all the channels available in the hopping sequence list to try to catch one of these advertisement packet, with the wanted PAN ID. Once the node receives such a message, it will synchronize to be part of the MAC layer network, and initiate the joining procedure at the network layer (or wait for a BNA, if the received message was only an EB and not a BNA).

The first message the joining node sends is a BNJR, informing the root node it wants to participate in the network. The root node replies with a BNTU, which purpose is mainly to inform the node that it has successfully joined. At this stage, the BNTU can only have the field "parent ID" set to 0, which is always the root node ID. The main goal of this message is to inform the node of its ID.

Once the node has received the first BNTU from the root node, it will consider it has successfully joined the network, and schedule a timer for starting to send BNTI messages.

Every time the root node receives a BNTI message from a node, it sends a copy of it to the network manager, in a NMI message. Then, the network manager processes and stores this data. The network manager also has an internal timer, and considers updating the topology (and schedule), every 20 seconds. If such a decision is made, the network manager sends a NMU packet to the root node, containing all the information about the new topology and schedule. It is then the responsibility of the root node to dispatch this information to the nodes with BNTU messages.

5.6 TOPOLOGY AND SCHEDULE UPDATE SEQUENCE

Using again the example presented in Section 4.2.2, and in Figure 4.1 in particular, here is presented what would happen after an update decision of the network manager for this example, and thus the reception of a NMU message by the root node, in Figure 5.5 and 5.6. In this example, it is assumed that the nodes are in a star topology in the initial state, and no timeslot have been allocated yet. Actually, the nodes just have a minimal slotframe with only one shared slot, in this initial stage. This is the typical kind of situation that happens right after the network has been started.

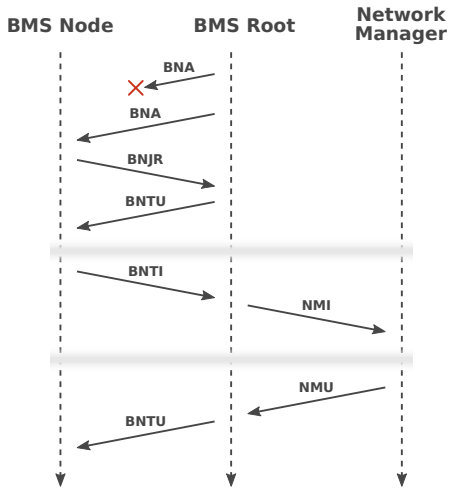


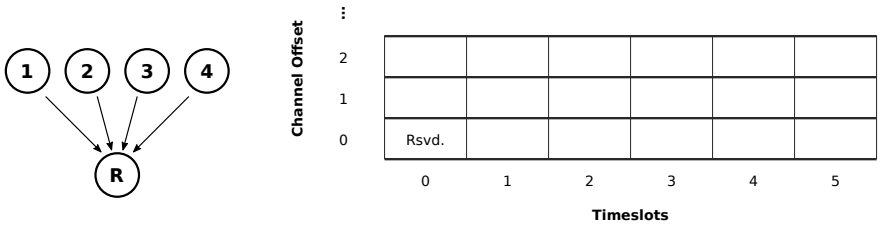
Figure 5.4: BMS with wireless communication node join sequence example.

Please note that BNTU messages only contain slotframe information about what the receiving node should do. Which means the slotframes that are presented, showing a global overview, are just here for illustration purposes.

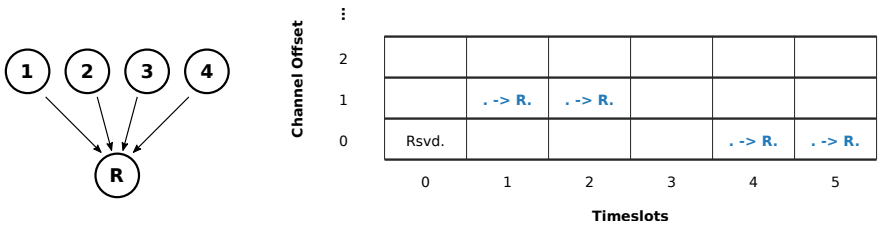
The nodes that are being sent BNTU messages to first are nodes that have rank 1 in the new topology. The order in which these nodes are sent BNTU messages to does not matter, and is implementation specific. To show this more clearly, the way it is done by the root node is presented in Listing 5.1. It matters to update rank 1 nodes first, in order for them to be ready when their children start sending them packets.

```
static bms_node_info_t * nodesListGetNextNodeWhichNeedsTopologyUpdate() {
        // First we look for nodes that should be rank 1, if one of them needs an
        // update
        for (uint8_t i = 0; i < BMS_NUM_NODES; i++) {
                if (nodesList[i].is_set) {
                        if (!nodesList[i].topology_info_is_fresh) {
                                if (nodesList[i].parent_id == 0u) {
                                        return &nodesList[i];
                                }
                        }
                }
        }

        // If we did not return anything yet, we just look for a node that needs to
        // be sent a BNTU to
        for (uint8_t i = 0; i < BMS_NUM_NODES; i++) {
                if (nodesList[i].is_set) {
                        if (!nodesList[i].topology_info_is_fresh) {
                                return &nodesList[i];
                        }
                }
        }

        return NULL;
}
```

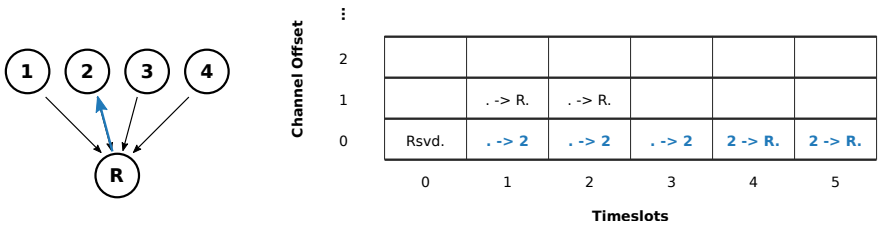
Listing 5.1: The function used by the root node to determine to which node it is going to send a BNTU message to next.



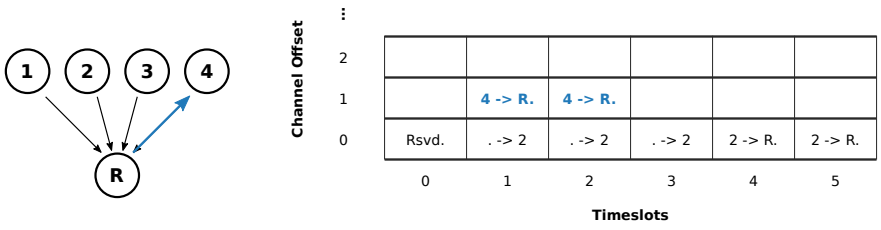
(a) Initial state.



(b) Right after the NMU has been received, the root node schedules its receive timeslots for the children it is going to have in the new topology.

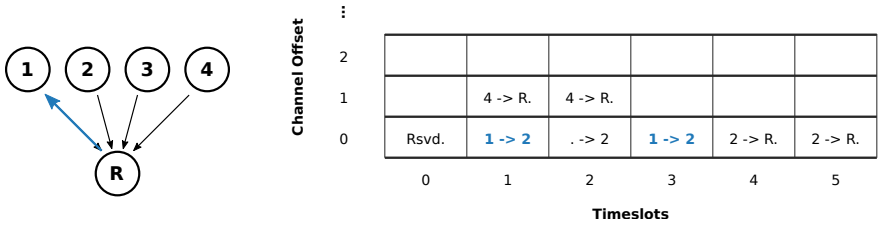


(c) First BNTU is sent to node 2. Here, not only the node 2 adds transmit timeslots to the root to its internal representation of the slotframe, but it also allocates receive timeslots for the nodes that are soon going to be its children.

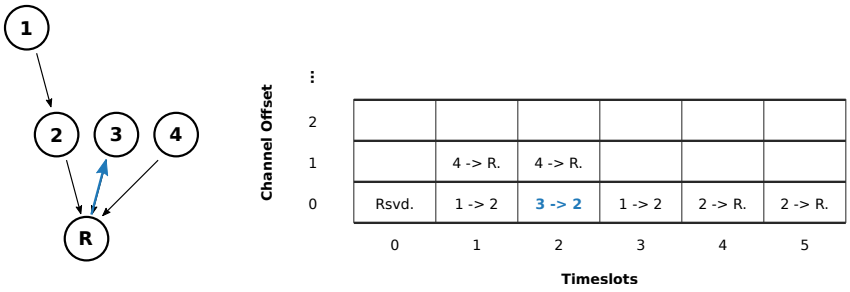


(d) Second BNTU is sent to node 4, which adds transmit timeslots accordingly, but will not have any children, so no receive timeslot is scheduled.

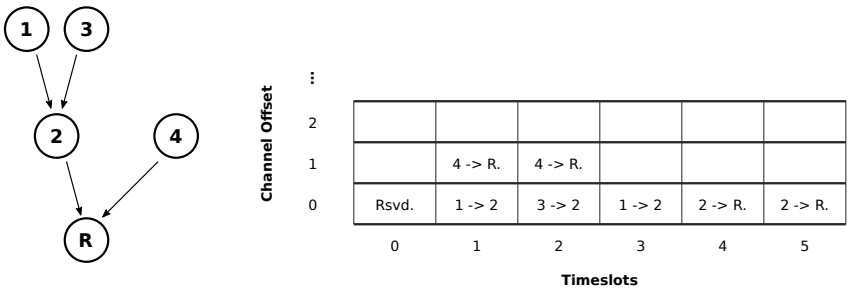
Figure 5.5: Network update with BNTU example: topology and schedule. Blue arrows represent BNTU messages, and highlighted blue timeslots represent newly allocated timeslots following this BNTU reception. This Figure is showing the first part of the process: updating rank 1 nodes.



(a) Third BNTU is sent to node 1, which is going to be a child of node 2 in the new topology.



(b) Fourth BNTU is sent to node 3.



(c) After all nodes have received their topolgoy update messages successfully, the topology and schedule do match the decision that has been made by the network manager.

Figure 5.6: Network update with BNTU example: topology and schedule. Blue arrows represent BNTU messages, and highlighted blue timeslots represent newly allocated timeslots following this BNTU reception. This Figure is showing the second part of the process: updating rank 2 nodes.

5.7 THE REPAIR MECHANISMS

Even though many repair mechanisms could be envisioned, as described in Section 5.9, only those which seemed strictly necessary have been implemented. They have been made to handle the case in which a node, or the root node and network manager, are reset.

If the root node is reset, it will start to send EB frames again. Any node trying to participate will see them in the end (using the mechanisms already implemented in Contiki-NG), and rejoin at the MAC layer. As the node was part of the older version of that same network before, it will continue to send BNTI messages every 10 seconds approximately. The root node will take a BNTI from an unknown node as a BNJR, and forward the information to the network manager. It will soon send a BNTU to this same node, as if a BNJR was received, to resynchronize with it for both topology and slotframe.

If a node is reset, it will start listening for BNA messages and use the regular joining procedure to become part of the network again.

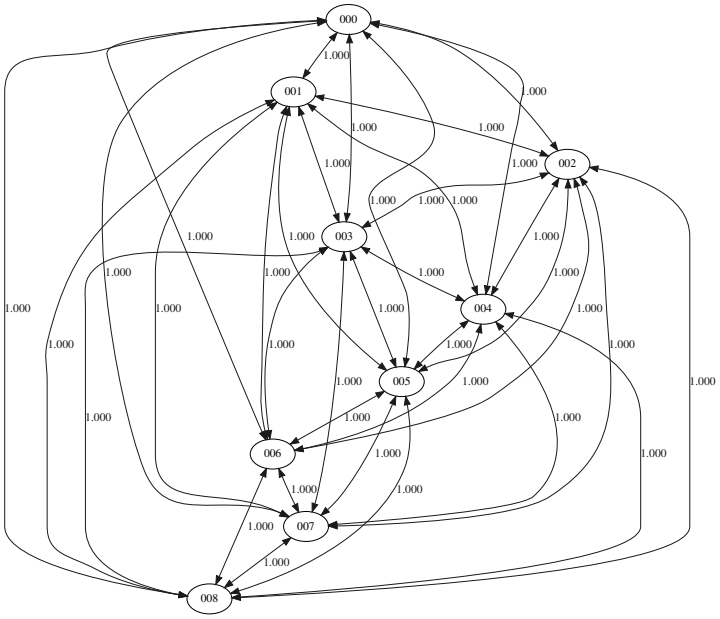
5.8 EXPERIMENTAL PERFORMANCE EVALUATION

In order to evaluate the behavior of the protocol, we tested it with a network of 8 CC2650 I3Mote nodes, each of them powered by a coin cell battery, and running our `bms-node` firmware, plus a route node, made of a CC2650 I3Mote with its daughter board, connected to a PC, running our border router firmware. The nodes were placed within the battery pack as shown in Figure 5.7. As the number of nodes was relatively small, and to make the joining procedure faster, we used the `TSCH_HOPPING_SEQUENCE_4_4` of Contiki-NG, which is `{ 15, 25, 26, 20 }`. It contains the 4 IEEE 802.15.4 channels that do not overlap with the three mostly used Wi-Fi channels, as described in Chapter 3.

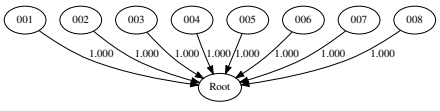
The strategy used by the the network manager is the one based on SD, and the parameters used are those described in Chapter 4. The maximum acceptable application PER for any path has been set to 10^{-6}, which means we accept to loose one packet per million on average.

During the test, the network manager outputs the results to a log file. When an update decision is taken, it also creates images containing: a graph representing its knowledge of all the links, the new topology, and the new slotframe. With this tool, it is possible to follow the evolution of the network over time. The initial state is displayed in Figure 5.8. As no data has been received, all links are assumed to be 100% PDR. By setting links for which we have no data to 100% PDR, we are giving incentive to the system to select these links, which will actually generate data for them, that the system will be able to use in its next check for update.

Figure 5.7: Nodes positions during the final test with our custom protocol.



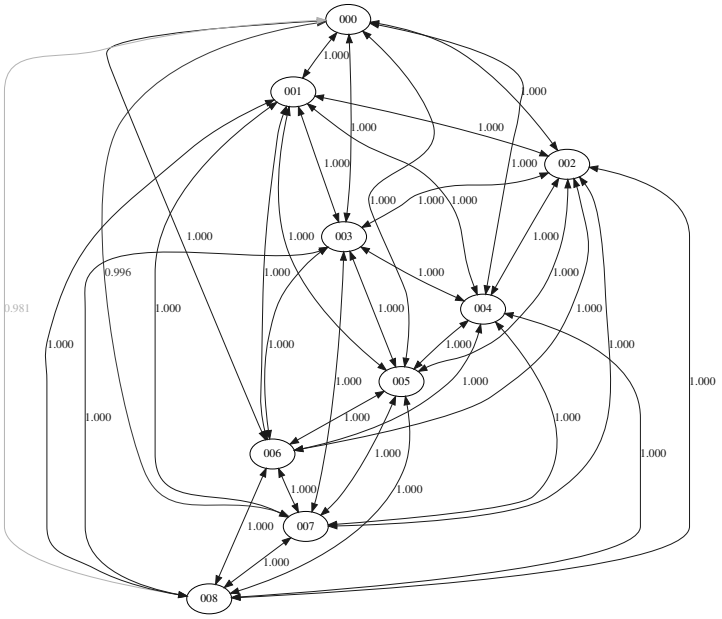
(a) Network manager knowledge of links PDR.



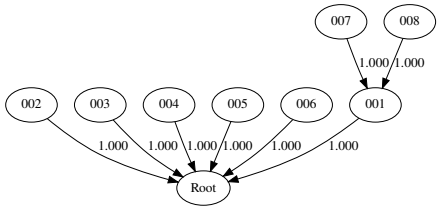
(b) Initial topology.

(c) Initial slotframe.

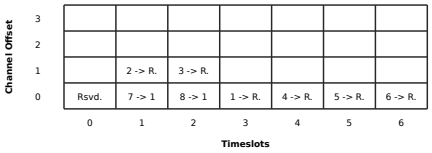
Figure 5.8: Final test: network manager knowledge of the links quality initial network topology and schedule. No data has been received from the nodes yet, so all links are assumed to have a 100% PDR.



(a) Network manager knowledge of links PDR.



(b) Topology.



(c) Slotframe.

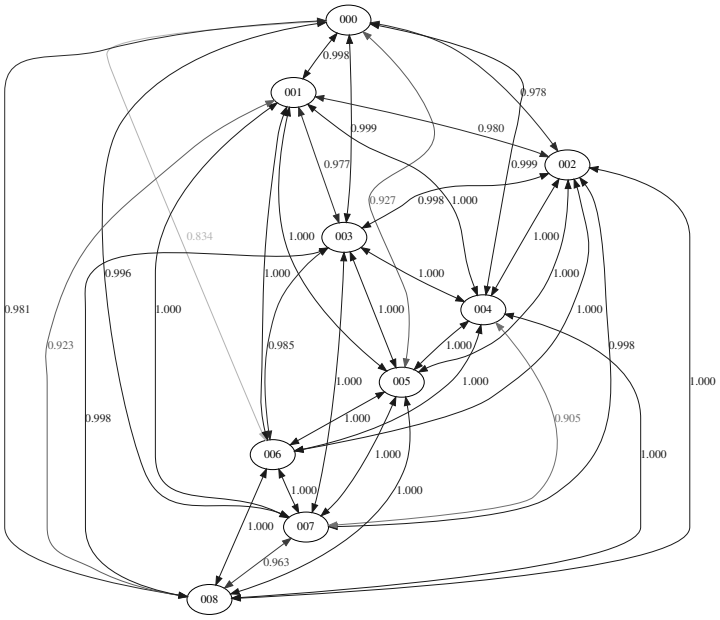
Figure 5.9: Final test: network manager knowledge of the links quality and first update decision, after 20 seconds, as some packets have already been lost.

After the first 20 seconds, some packets have already been lost, so the network manager makes an update decision. This situation is shown in Figure 5.9. It is visible that a few packets have been lost between nodes 7 and the root, and node 8 and the root.

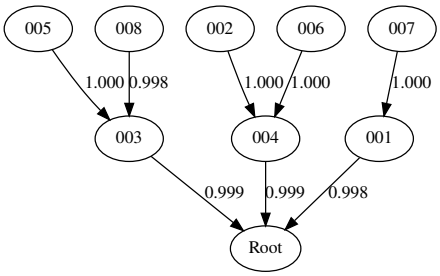
After letting the test run for one hour, during which 20 update decisions have been made by the network manager, the network was relatively stable: no update decision occurred in the last 6 minutes. The result is shown in Figure 5.10.

From the log file, we can see that 4 application layer packets have been lost in this last 6 minutes period, and occurred approximately 6 seconds after the update. So the losses may actually be due to the update. After this, no application layer packet has been lost until the end of the test.

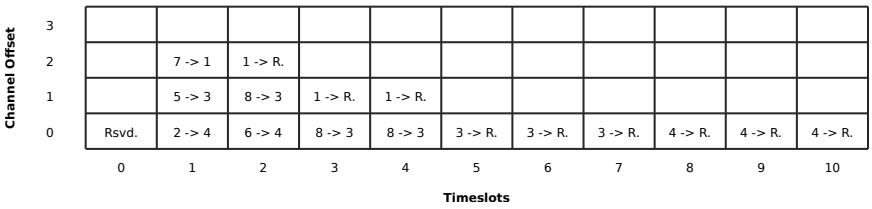
Going back to the actual layout of the nodes inside the pack, at the end of the test with the last topology, the nodes of rank 1 are physically closer to the root. This is illustrated in Figure 5.11. However,



(a) Network manager knowledge of links PDR.



(b) Topology.



(c) Slotframe.

Figure 5.10: Final test: network manager knowledge of the links quality, and topology and slotframe at the end.

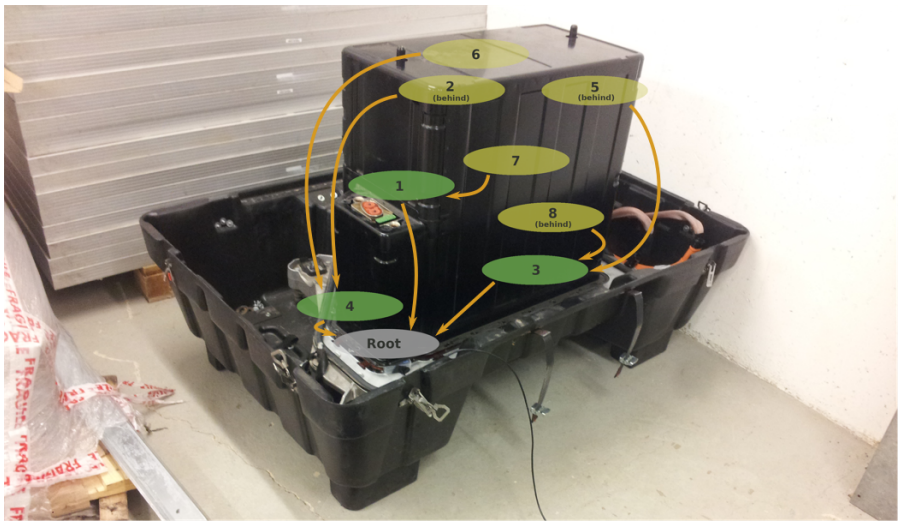


Figure 5.11: Network topology at the end of the test with regard to the nodes' actual location. Rank 1 nodes are in green and rank 2 nodes are in yellow. The arrows show the topology.

for rank 2 nodes, their parents are not always the closest in the rank 1 nodes.

In order to have an idea of when the update decisions occurred during the experiment, Figure 5.12 has been made. Many updates occurred at the beginning, which was expected, because some links that are tried can be of very low quality, but also because there is not much data yet, so any packet lost has more weight than when there is more data. What is more surprising is the number of updates occurring between 40 and 50 minutes. The most obvious explanation would be that some losses are due to the topology, or rather schedule, update. For example if a node that should be of rank 2 in the new topology receives its BNTU message before the node that should be its parent in the new topology, it will configure its dedicated transmit timeslots and try to send packets, while the corresponding receive timeslots have not been configured yet by the new parent.

Another explanation would be that a node of rank 1 can receive its BNTU a few seconds before its future child receives its own. This would mean that the node which will be of rank 2 in the new topology will still send its data packets to the wrong node while the topology and schedule update is ongoing, and especially using the transmit timeslots for which the parent in the old topology has discarded its receive timeslots in its local representation of the schedule. What is clear here is that topology updates cause packet losses. And the obvious reason is because updates take time. During these update periods some nodes use the old topology and schedule, while others use the new ones. As the nodes continue to send data packets during the update process, losses will occur, be taken into account in the network manager, and make it have a representation of some links

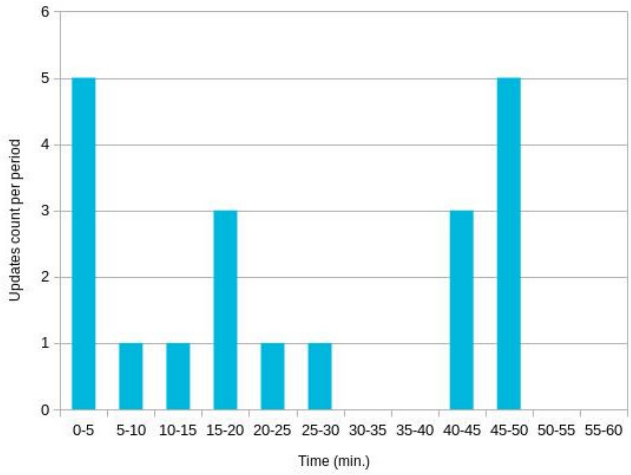


Figure 5.12: Times in which topology and schedule updates occurred.

that is worst than what they actually are. This would at least explain that a period of instability can follow an update decision. Finally it is worth mentioning that the links quality are very high, and every loss will have a big impact on each link average PDR.

It is probably not desirable to stop sending data packets during the time of the update, as this would disturb the main BMS process in the MCU. However, a way to try to solve the problem discussed here would be to stop counting the losses during the period of time it takes to do the update. This could be done in the following way. When the root node receives the update decision in a NMU packet, it would set a flag that would mean "stop counting the losses" in the BNA packets, and wait for 5 to 10 seconds before starting to send the BNTU messages. After all the BNTU messages have been sent, the root node would wait for 5 or 10 more seconds before unsetting the flag in the BNA messages. This could be a way to strongly reduce the number of counted packet losses due to an update, thus improving the stability, at the cost of slightly delaying the start of the update procedure.

A simpler way to try to manage this problem could be to forbid update decisions in the network manager for 1 or 2 minutes after one has been taken, in order to have more samples for the new links that are used, and give less weight to the losses that can be related to the update process.

Sadly, the experiment described here was conducted only once due to a lack of time. More data would help to better understand the phenomena taking place here, and how to improve the protocol.

5.9 DISCUSSION AND FUTURE PERSPECTIVES

The proposed protocol can efficiently build a network topology, and provide enough timeslots for retransmissions in such a way that the

network is made very reliable, as is presented in Section 5.8. However, it also has downsides, and there are many ways to improve it.

First of all, most values have been chosen arbitrarily. BNA messages are sent every 3 seconds approximately, and the "check for update" method of the network manager is called every 20 seconds. These are parameters that have a huge influence on how the network behaves, and the value they should have should be carefully studied.

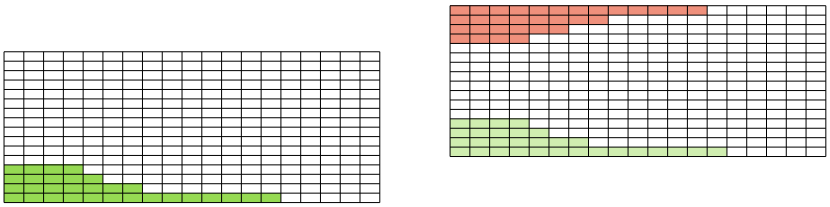
There is also room for improvement in the way the link quality data is managed and the per link PDR is calculated. With the current system, if a link is tried and suffers from many losses, the system will not go back to it and try it again. Although it is possible that the losses are due to interference that occurred with the vehicle in a specific location, and that the data does not reflect the actual link quality. Maybe a feature that makes the network manager "forget" the data after some time could help here. Or a feature that forces the network manager to retry the "bad" links after some time, to verify that a link evaluated as weaker is indeed not very good.

The current update criteria is the difference between the current topology value according to the objective function, and the value of the possible new topology. This difference is then compared to the last four differences, and if it is bigger than all the previous differences, the update decision is taken. This a way to both have exploration of the different links, but also some stability. There is obviously a trade-off here, and how to find the balance between these two behaviors should be studied.

The main problem is currently when a node receives a schedule update it will remove all timeslots from its slotframe and allocate the new ones. However, when this happens a child node may still have transmit timeslots allocated and still use them, which can cause packet losses. A non-destructive, or rather less destructive, mechanism could be employed, where a node keeps its receive timeslots in its slotframe for some time after receiving an update, or at least the timeslots that do not conflict with the newly allocated timeslots. To push this idea even further, it would be possible, without altering the path quality, to alternate between starting to allocate the channel offsets from the smallest offset and then the biggest. This would be to limit the possibility that the old and new timeslots overlap. This idea is illustrated in Figure 5.13.

5.10 CONCLUSION

In this chapter, we proposed a new protocol for managing a WSN for a BMS application. We described the topology and slotframe update mechanisms, as well as the control packet content, the join procedure, and repair mechanisms. We also presented our experiment inside a battery pack, using the implementation of this protocol that we did in



(a) Slotframe example. Allocated timeslots are in green. Timeslots are allocated starting starting with channel offset 0.

(b) New slotframe. Timeslots are allocated starting with channel offset 15. Timeslots in red are the timeslots in the current slotframe. Timeslots in light green are the timeslots of the previous slotframe.

Figure 5.13: Timeslot allocation idea: reducing the number of overlapping timeslots in an old and new slotframe.

Contiki-NG. We finally discussed the result and gave ideas to further improve the protocol and network management strategy.

The presented system is still a proof of concept, and some features are missing before it is suitable for deployment and used inside an EV. For example, the network manager would need to record the link quality data history, to be able to reuse it after a sleep period, when the vehicle is started again.

However, the system has proven to be functional, the network manager has successfully managed to optimize the topology and schedule in order to obtain a high QoS. Also, the experiment we did validates the algorithm and management strategy presented in Chapter 4. In the next chapter, we are going to conclude on this work and give perspectives on what further research on this subject could be.

6 CONCLUSIONS AND PERSPECTIVES

6.1 CONCLUSION REMARKS ON THE PRESENTED WORK

The BMS of an EV is a key component, and a safety oriented device which is absolutely necessary. It has to supervise the battery cells to guarantee they will be operated in their safety window, and to estimate their SOC. In large battery packs, like in these which can be found in cars, the BMS is a distributed system. The communication between its submodules must be reliable, and guarantee a high QoS to the system. This work is about trying to change the wired communication links to wireless. This is why we reviewed the state of the art protocols for wireless communications, studied the EV environment with experimentation, and proposed a new management strategy and protocol to achieve this network path quality objective, and make sure the information is delivered to the MCU on time.

In Chapter 2, we started by studying more in depth the protocols and messages used in today's implementations of BMS using wired communication links. We identified that the information which matters the most are the messages from the CSUs to the MCU containing the voltage and temperature data. Then, we reviewed some state of the art protocols for wireless communication, and we set our choice on IEEE Std. 802.15.4-2015 with TSCH mode. We based this decision on preexisting works on communication within a vehicle environment, and in a battery pack especially. We also reviewed some topology and schedule management techniques available in the literature, and explained why we could find one that is more efficient in our specific use case.

Then, in Chapter 3, we tested the behavior of IEEE 802.15.4 within a battery pack environment. We discovered that the link quality depends on the position of nodes. Also, that the pack enclosure has an impact on communications and seems to make them better. What does make them worse however is interference with other users of the 2.4GHz band, and especially Wi-Fi. A way to solve this problem is temporary blacklisting of the problematic channels. More information on this is available in [61]. But, most of all, we saw that links quality is high within a battery pack environment.

Based on the results we obtained, in Chapter 4, we proposed two new centralized network management strategies. The first one, based on the LP technique, has proven to be efficient in finding an optimal topology based on link PDR information. The second one, mostly based on the SD technique, gives results that are almost as good for

both topology management and scheduling. As it is lighter in terms of computing time, it is the method we have chosen. Also the fact that it is faster and requires less memory makes it more appropriate for being run on an embedded device. In the end of the this chapter, we try several parameters for the SD technique, and select the values that seem best.

Finally, in Chapter 5, we proposed a new protocol for centralized network management, and to propagate efficiently the routing and schedule information to the nodes that participate in the network. This protocol is based on periodic messages and asynchronous events, rather than acknowledgments. We implemented this protocol in Contiki-NG, and tested it inside a battery pack. With this experiment, we could verify that our strategy is efficient, and we were able to achieve a maximum path PER of $10^{-}6$ %, while having a slotframe that has only 10 dedicated timeslots to convey the voltage and temperature data to the MCU.

6.2 FUTURE WORK AND PERSPECTIVES

The system we proposed here for BMS with wireless communication is not sufficient by itself, and some other aspects need to be worked on and investigated before the technology could be suitable for deployment. These subjects are discussed in this section.

6.2.1 *A word on security*

Deploying such a network, for the safety critical device that a BMS is, must be done with security in mind. In this section we present some of the problems and attacks that may occur in the case of BMS with wireless communication, and for some of them we provide some ideas to implement countermeasures. However, a complete security review is out of the scope of this document, and such a study should be done before any deployment can be envisioned.

6.2.1.1 *At physical layer*

Switching from a wired to a wireless communication medium for battery management means that this new medium is shared. Also, as the metallic enclosure of the pack contains holes and is not fully closed, it means the electromagnetic waves used for communications may go in and out the pack. This is also what has been measured in Section 3.4.4. So, an adversary node may use this property to interact and potentially interfere with the network.

The first kind of attack that comes to mind is Denial of Service (DoS) [68, 69], meaning an attacker would broadcast over the channels used by the network to jam the communications [70]. It is well known

that if the attackers will succeed if they use enough power to cover the whole band used by the network.

If the attackers cannot consume this amount of power and focus only on one or some of the channels in the 2.4GHz band, then using TSCH is a way to mitigate this kind of attack. Either the attackers will jam only a subset of the channels used by the WSN, or they would need to acquire knowledge about the TSCH schedule and determine a hopping sequence allowing them to disturb the network.

6.2.1.2 *At MAC and upper layers*

The BMS is a safety oriented system, and the ability for the MCU to have an accurate and reliable representation of all the battery data must be guaranteed. Regarding this statement, some kinds of attacks have been identified. They are listed below.

AUTHENTICATION It is required that the MCU can trust the data received from the nodes participating in the WSN. Therefore, it is not acceptable if an adversary node uses Spoofing to impersonate a legitimate node and sends wrong values to the MCU.

To deal with this possible attack, it should be required from the nodes to authenticate before sending data to the master.

INTEGRITY It is important that the data received by the nodes, and especially the MCU, are not modified along the way. Other threats in this category include packet injection, falsification, and replay attacks. The latter consists of an adversary node overhearing a packet and replaying it after some time, possibly in a periodic way, to corrupt a node's data.

On reception, data integrity should be verified using a checksum.

TRACEABILITY BMS nodes, or at least the MCU should keep track of the history of the battery data over time, and the actions taken, in a log file. These may be useful in development, or if a fault occurred to investigate and determine the cause. It is important that the system is designed in a way that does not allow corruption of this log file.

CONFIDENTIALITY Another risk is data about the system being leaked to other parties through an eavesdropping node. Confidentiality means an unauthorized node can not read the data clearly. If the confidentiality of topology and schedule related messages is not guaranteed, this may increase other weaknesses in the network.

To mitigate this risk, packet encryption should be used, at least at the MAC layer, and if possible at the transport or application layer.

AVAILABILITY Availability of the most recent data is critical for the MCU, as it relies on it to detect and react to any fault. Attacks aiming

at reducing the network availability include flooding, distortion of resources, amplification and reflection. This kind of attacks would usually result in flooding the MCU node with a very large amount of packets.

To reduce the risks of the attacks described here, bi-directional authentication is probably a good base for a solution.

6.2.2 *Sleep mode, wake-up procedure, and sleep power consumption*

A key aspect of BMS implementation is its power consumption when it is unused. It is highly undesirable if the nodes empty the battery if the vehicle is parked for a long period of time. The Lithium-Ion batteries have very low self-discharge, and the users will expect that the pack has more or less the same SOC if the vehicle has been unused, even for a few months.

In a CSU board with wireless capabilities, the consumption of the radio part is the highest when the node is either transmitting or receiving. For a CC2650 chip, the current consumption rises up to 10mA during these periods of time, whereas the consumption when the chip is only running code is close to 1mA, and can go as low as 1µA when the chip is in standby mode [82]. The analog-front end part of the board can be set to sleep mode through the digital bus between the radio chip and the analog-front end, and woken up through the same channel.

For these reasons, the nodes must go into sleep mode when the vehicle is parked. But then the problem is to wake them up rapidly when the user turns on the vehicle. The wake-up procedure of the BMS will probably be triggered by a button on the dashboard of the vehicle, or at least a system which is external to the battery pack. This wake-up command would then be received by the MCU, most likely over the CAN bus, which role will be to propagate it rapidly to the CSUs.

A first approach to deal with this problem could be to keep the network always on, and use a dedicated slotframe for the "sleep mode" that would be very long and containing only one shared slot (or a slot that would be a receiver slot for all the CSUs and a transmit slot for the MCU). The MCU could then use this shared slot to broadcast a frame containing the wake-up command. This concept is referred to as "duty-cycling MAC" [71], because it consists in using a duty-cycle at the MAC layer to not keep the radio always on. One of the difficulties in this approach is to send enough packets during the sleep period to keep the nodes synchronized. Indeed, in a TSCH network the clock of the different nodes drifts over time, and they use the frames they receive to remain synchronized.

A more advanced way to approach this problem is to use Wake-Up Radio (WuR). This consists in using a dedicated circuit for the

idle-listening time, which is capable of being addressed by a wake-up packet, and to wake up the main radio circuitry [71]. WuR is targeting applications that are delay sensitive and event-driven, while decreasing the power consumption of the node compared to a duty-cycling MAC. One of its advantages is that it can be kept always-on. The downside of WuR is that it requires extra hardware on the board to send and receive the wake-up packets, and trigger a wake-up interrupt into the main radio microcontroller. Some options to do this exist in the literature that are not based on radio-frequency, but on optical or acoustic waves instead.

Considering BMS with wireless communication requires extra hardware compared to using wired communication, and WuR requires even more hardware and space on the CSU boards, duty-cycling MAC is probably the best approach here.

6.2.3 *Battery second life applications*

One use case that may benefit from wireless communication for BMS is battery second life and battery reconfiguration. When the cells used in a vehicle have a SOH below a given threshold, usually set around 75% [24], they are not as useful as part of the powertrain of an EV. But the modules can be reconfigured into another pack and used as stationary storage.

Wire reduction is interesting in this scenario, because it makes pack reconfiguration and handling the modules easier. However, it is not likely that this scenario will happen many times in a cell's lifetime.

What could be interesting here is that having a microcontroller on each CSU also means the cells data may be stored locally. And if the CSU are kept as-is in the new stationary application, the data they would contain can be reused for estimations in this application as well.

Although for this to be possible and be actually useful, it would be necessary to standardize how a CSU accesses the network and makes its data available. This is required for any manufacturer of stationary storage pack to be able to reuse the modules from an EV pack without having to modify them.

6.2.4 *Potential interest of the industry for wireless communication for BMS*

Even if determining if using wireless communication between BMS submodules is not the purpose of this work, in this section we will give a few elements trying to foresee if the industry will adopt this technology. We are going to discuss a few aspects, and highlight some upsides and downsides of the technology.

6.2.4.1 *About weight reduction*

First of all let's take an example about weight reduction. A Renault Zoé battery pack measures $1280 \times 1630 \times 335$ millimeters [8]. Let's assume there are 6 battery modules on each side (we know it has 12 total). The maximum space between each module, and hence the maximum wire length between two modules is:

$$L_{max} = \frac{1630}{5} = 326 \text{ mm} \tag{6.1}$$

To make a differential pair between two CSUs with wired communications, it is necessary to have twice that length of wires, which is 652mm. If we assume the Standard Wire Gauge (SWG) of the wires used is 20, which is an average value (they may actually be thinner), we can calculate the weight that would be removed between two modules, knowing that 100 meters of wires of this size weights 589 grams[1]:

$$W_{\text{removed wires}} = \frac{0.652 \times 589}{100} = 3.84028 \text{ g} \tag{6.2}$$

So the maximum weight of wires that could be removed between two battery modules, by switching to wireless communication, is around 4 grams. Now, let's look at how much weight wireless communication adds to the system. In Appendix A, we built a board of size $200 \times 100 \times 1$ mm. The weight of this board, which is relatively standard with a copper height of 35 µm, is 77 g, with no component soldered on it. If we look at a CC2650 Launchpad, and measure the surface occupied by the CC2650 chip, its bypass capacitors and the antenna (which is probably slightly less than the required area for wireless communication components on a CSU board), we obtain:

$$S_{wireless} = 10 \times 20 + 10 \times 30 = 500 \text{ mm}^2 \tag{6.3}$$

The area we considered for this can be seen in Figure 6.1.

From this, we can calculate the weight added to a CSU board by adding the wireless communication circuit:

$$W_{\text{added wireless}} = \frac{77 \times 500}{100 \times 200} = 1.925 \text{ g} \tag{6.4}$$

In the Renault Zoé, there are 12 modules, so 12 CSU boards, and a MCU, so the weight added by wireless communication is:

$$W_{\text{added wireless total}} = 13 \times 1.925 = 25,025 \text{ g} \tag{6.5}$$

And the weight removed thanks to wireless communication is:

$$W_{\text{removed wires total}} = 12 \times 3.84028 = 46,08336 \text{ g} \tag{6.6}$$

1 From <http://www.mtlexs.com/technical-specifications-details/40/weight-of-copper-wire>.

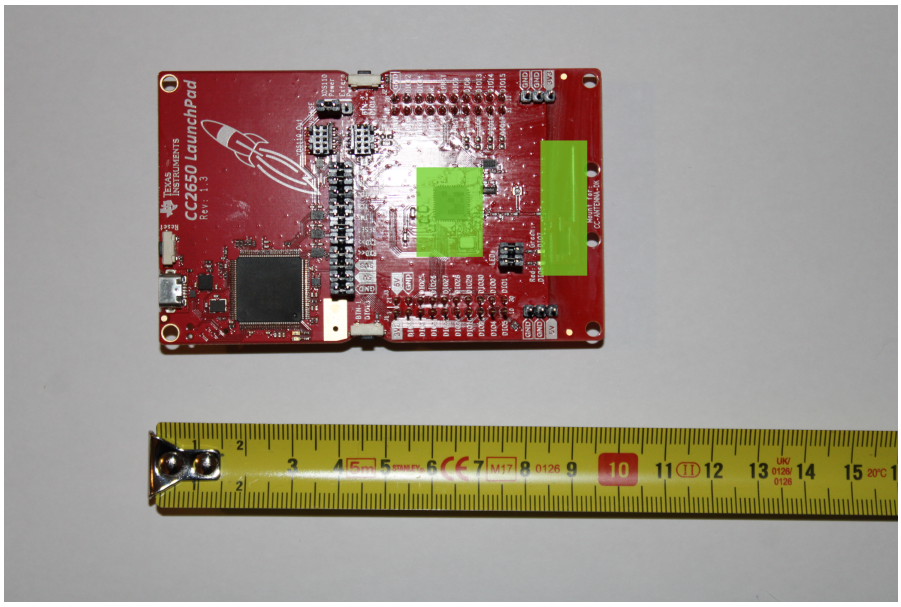


Figure 6.1: A Launchpad CC2650 board. The area shown in green is the area we took into account to calculate the minimum board surface that need to be dedicated to wireless components.

To conclude, in the case of a Renault Zoé battery pack, which weights 290 kg, the total weight reduction to be expected with wireless communication is approximately 21 grams, and this is just a rough estimation, favorable to wireless communication as we did not consider the weight of the electronics parts or solder paste.

6.2.4.2 *About assembly procedure*

While removing wires used for communications greatly simplifies the assembly procedure of the EV's battery pack, it also adds an extra step. When the CSU or MCU boards are made, after the pick and place machine has placed the components, and the board has been baked into a reflow oven, the microcontroller used for communications needs to be programmed. This is a common step to have on a board assembly line, that can be done automatically with a machine, but is still an extra step.

6.2.4.3 *About hardware availability*

This manuscript has been written during the Covid-19 pandemic, which slows down production in many sectors of the world's economy. Especially in the field of electronics, where some chips, especially microcontrollers, suffer from a penury. Introducing wireless communication in EVs BMS means adding extra dependencies on hardware components to be able to build the final product.

6.2.4.4 *About complexity*

In section 1.1.2.3, we discussed about complexity in BMS, about both hardware and software level. With BMS with wireless communication, the BMS software necessarily needs to be distributed. This makes it harder for developers to test and debug it. Having more hardware components also increases the number of potential failure points. These aspects could also be an extra cost of BMS with wireless communication to consider.

However, maybe car manufacturer will find advantages in having the software distributed, for example to manage locally the cells balancing decision. This also gives them a tighter control over sleep mode management, as they could perform operations within the CSUs while the MCU is sleeping, should this be a desired behavior.

6.2.4.5 *About cost of hardware*

As of today, the public price of a CC2650 chip is around 2$[2]. Just as an example, some copper wire may be available at 0.33€ per meter[3], which makes BMS with wired communication apparently cheaper, at least on the hardware side of things.

6.2.4.6 *Speculations on BMS with wireless communication adoption by the industry*

What has been exposed in this section are just a few elements to open reflection. There are multiple reasons why the industry may or may not adopt this technology, and not all of them can be known from the outside. However, we can see that even if some upsides of BMS with wireless communication are real, there are also some significant downsides. And even considerations that seemed to be *a priori* good reasons to make the change are not that evident, like weight reduction. Hence, it is impossible to tell what big car manufacturers will do in the future, and adoption of BMS with wireless communication is not obvious.

6.2.5 *Going further with wireless communications for BMS*

As we have seen in this work, it is possible to implement BMS with wireless communication and make it reliable, at least from a network perspective. However, there are many remaining open questions on the subject. First of all, there are many ways to implement it. We proposed a solution that involves multi-hop and routing. Even though we obtained good results with it, maybe a solution relying on star topology can be sufficient. Furthermore, if routing is employed, there may be

2 From <https://www.ti.com/product/CC2650#order-quality>.
3 From <https://fr.farnell.com/alpha-wire/3053-vi005/fil-vlt-20awg-10-30awg-30-5m/dp/2290840 >.

other ways to implement it. And even about our solution, based on SD for managing the topology, it is possible to push it further. Methods derived from it include "Deepest descent" and "Multistart descent", which increase the chances of finding an optimal solution to the problem, by trying to go one step further when a local minimum is reached, or by trying to use multiple initial solutions, respectively. Moreover, our solution focuses only on the most important traffic pattern: uplink messages to the MCU containing voltage and temperature data. But there are also other messages exchanged, like balancing commands or requests for auto-test from the MCU to CSU. These traffic patterns should be further studied and may require allocation of more timeslots in the slotframe.

As discussed above in this chapter, we covered only network management and reliability aspects, even if BMS with wireless communication is a much wider subject. Subsequent studies should focus first on security, and sleep mode and wake-up scenario, which are prerequisites for making this technology usable.

What we proposed here is for a specific application, in a very specific environment. Maybe it can be used as a basis for other WSN applications that would benefit from centralized management. Or it is possible that the techniques used, especially the one based on SD, can be used or extended in another wireless network context. The first one that comes to mind is micro-grids deployment, where small production and stationary storage devices are deployed in a relatively small area, and can benefit from synchronization between each other. There, the physical size of the links is more important than in a battery pack, and wireless communication may be more appealing. Although in this field Power Line Communication (PLC) is a serious competitor.

Finally, there is no certainty about the adoption of BMS with wireless communication by the automobile industry. Even if some upsides can be foreseen, some downsides exist as well. The future of BMS with wireless communication is uncertain, and leaders of big car companies will have to rule on whether or not its downsides are outclassed by its advantages.

A

APPENDIX — SIDE PROJECT: BUILDING A BMS BOARD FOR A SMALL BATTERY PACK

A.1 OBJECTIVE

During this thesis, and as a side project, I decided to build my own BMS board. The goal was to better understand what a BMS actually is, and feel what matters in the design of such a system. As I wanted something I could potentially use one day in my own battery-powered application, I went towards something quite small and standard: a BMS to monitor a 12 cells series Lithium-Ion battery pack. What inspired me in doing this is the VESC project[1], from which I learned a lot.

A.2 DESIGN

I decided to go with the `bq76940` for the analog front-end ASIC. As I wanted to limit the risk and use something I know for the MCU, I chose to use the same microcontroller as in the VESC project, the `STM32F405RGT6`.

The design of the hardware systems is directly derived from these two main components' datasheets. The conception of the board has been made in Kicad[2]. The top level sheet of the schematic is visible in Figure A.1.

A.3 MANUAL ASSEMBLY

A.3.1 *The board itself*

The board is a 10 by 20 centimeters Printed Circuit Board (PCB). It is visible in Figure A.2 and A.3. In these images, only the `bq76940` has been soldered.

A.3.2 *Soldering*

I chose to do manual assembly, which has been a great learning experience. With flux and patience, it is actually possible to solder the components that have a 0.5 mm pin pitch, even with lead-free tin. A

1 VESC project website: <https://vesc-project.com/>
2 Kicad's website <https://www.kicad.org/>.

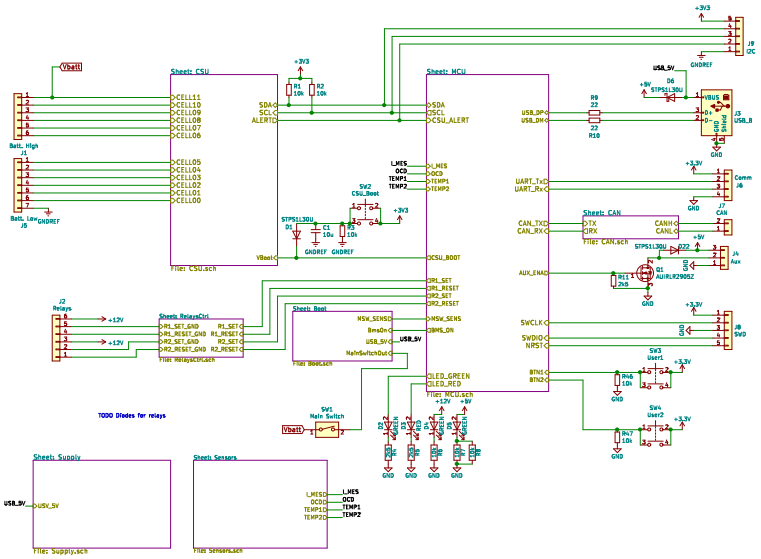


Figure A.1: Custom BMS board schematic, top level sheet in Kicad.

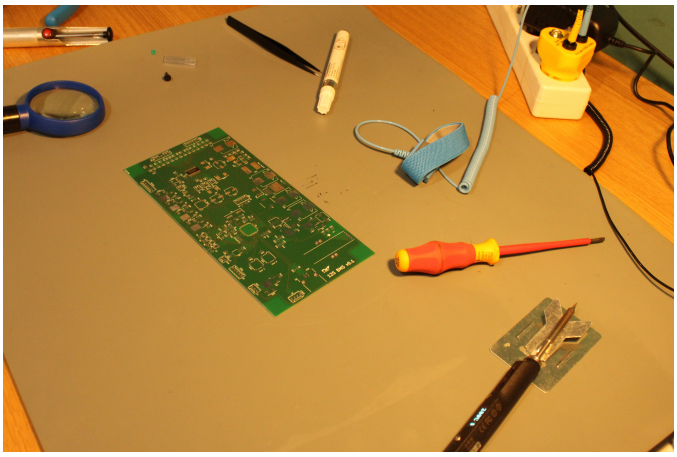


Figure A.2: Custom BMS board, mounting the bq76940 (1).

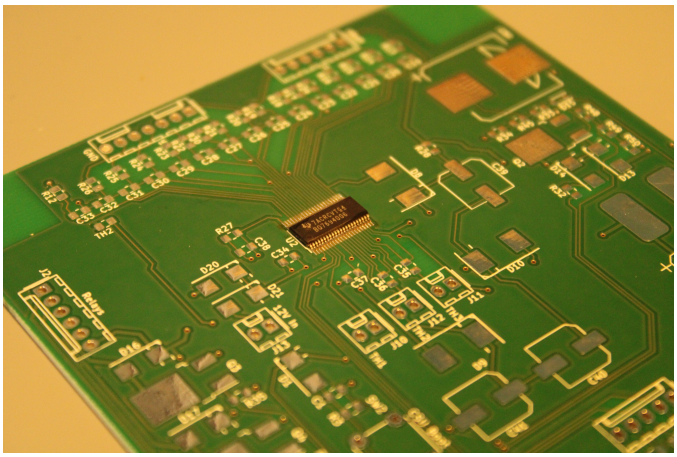


Figure A.3: Custom BMS board, mounting the bq76940 (2).

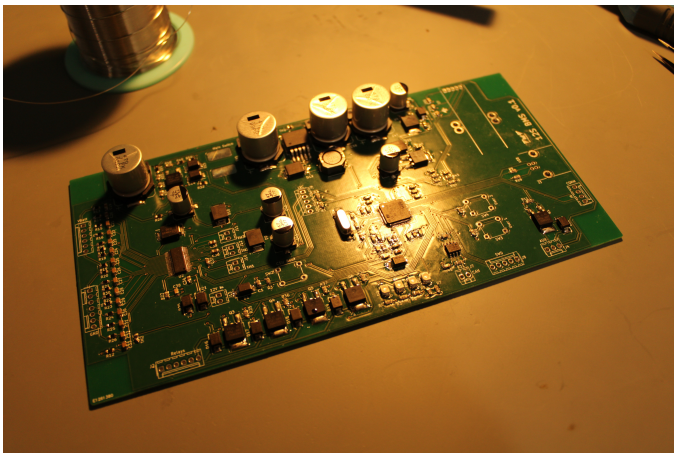


Figure A.4: Custom BMS board. Here most footprints have been populated.

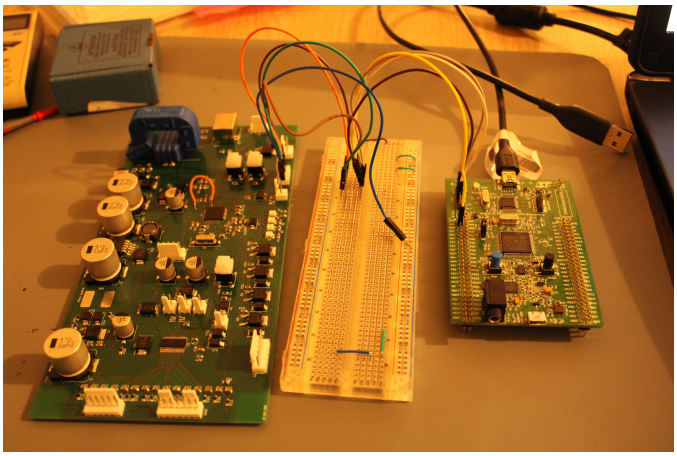


Figure A.5: Custom BMS board, ready to be programmed for the first time.

picture of the board in which most components have been soldered is shown in Figure A.4.

A.3.3 *Fixing mistakes*

When trying to flash the STM32 microcontroller for the first time, I realized it was behaving as if it was turned off. Actually some of the power supply pins were not soldered properly, which was easy to fix.

Also, while doing the schematic, I did not wire some of the software download pins properly. This has been fixed by adding a modification wire on the microcontroller. The fix is the orange wire visible in Figure A.5.

A picture of the board being ready to be programmed for the first time, and after assembly issues have been fixed, is displayed in Figure A.5. At this stage, I was able to run a basic Light-emitting Diode (LED) blink example.

A.4 CURRENT STATUS

Currently the hardware part of this board seems functional. I also implemented a simple PC version of the MCU software, using OCV and Coulomb Counting methods for SOC estimation. However, this piece of software still needs to be ported onto this embedded platform, and tested, before this side project can be called a success, and only then I will publish the design and software.

B

RÉSUMÉ LONG

B.1 INTRODUCTION

L'apparition des nouvelles batteries Lithium-Ion, ainsi que les enjeux climatiques [4], favorisent une nouvelles adoption du véhicule électrique. Ces nouvelles batteries, ont d'importantes densités énergétiques. Cela en fait une solution attirante pour stocker de l'énergie dans un véhicule. Elles sont aussi des systèmes sensibles dont la tension et la température doivent rester dans une plage bien définie pour garantir leur sûreté de fonctionnement [5]. Si la tension d'une de ces cellules devait descendre en dessous d'un certain seuil, elle serait détériorée. Si cette tension devait dépasser un seuil, la cellule peut prendre feu. Ces plages acceptables de température et de tension sont souvent appelées « fenêtre de sûreté ».

Une cellule batterie Lithium-Ion est un système complexe avec beaucoup de non-linéarités, et estimer la quantité d'énergie restante dans une cellule n'est pas trivial. En effet, la tension en circuit ouvert varie de manière non-linéaire avec l'état de charge [9]. De plus, la tension aux bornes d'une cellule varie en fonction du courant de charge ou de décharge [16]. Ce phénomène s'appelle l'hystérésis.

Ces phénomènes complexes ont aussi un impact sur la manière de piloter le processus de charge d'une batterie. Durant la charge, la première phase est pilotée en courant, et est appelée phase de « courant constant ». La deuxième est appelée phase de « tension constante », et à lieu à partir du moment ou la tension maximale du pack est atteinte. Au cours de celle-ci, le courant décroît alors lentement vers 0.

Le gestionnaire de batterie, en anglais *Battery Management System (BMS)*, remplit plusieurs rôles. Tout d'abord il supervise individuellement chacune des cellules de la série qui compose le pack pour s'assurer qu'elles restent dans les plages de température et de tension données par le fabricant. Donc le BMS est avant tout un composant augmentant la sûreté de fonctionnement de la batterie. Il limite le courant qui entre et sort de la batterie, soit en communicant les valeurs à ne pas dépasser aux autre équipements sur le bus CAN, soit en ouvrant un relai placé sur l'un des terminaux du pack batterie. Si le BMS échouait à faire cela, le pack batterie pourrait être détérioré ou prendre feu en cas de problème. Donc, si le BMS ne peut pas collecter ces données, il ne peut plus assurer un fonctionnent sûr du pack batterie, et le véhicule devrait cesser de fonctionner.

Le BMS est aussi le composant qui estime l'état de charge de chacune des cellules [13]. Pour fournir cette fonctionnalité, il mesure

de manière périodique la tension aux bornes de chaque cellule de la série. La période d'échantillonnage pour ces valeurs dépend de l'implémentation, mais est généralement de l'ordre de 500ms [15], [17]. La capacité d'un pack batterie décroît avec le temps et l'utilisation. Cette perte de capacité, appelée état de santé, est aussi un paramètre calculé par le BMS.

Dans un pack batterie, les cellules sont câblées en parallèle et en série. En parallèle, les cellules les plus chargées vont naturellement se décharger dans les cellules les moins chargées. En revanche, quand elles sont câblées en série, du déséquilibre peut apparaître entre-elles au fil de l'utilisation. Équilibrer les cellules de la série est donc aussi l'un des rôles du BMS. On trouve deux familles de méthodes dans la littérature [22], [23] : l'équilibrage passif, où les cellules avec la tension la plus élevée sont déchargées dans une résistance, et l'équilibrage actif qui peut être implémenté de diverses manières. L'équilibrage passif est celui qui est utilisé le plus souvent, car il est plus simple à implémenter, et requiert moins de composants.

Enfin, le BMS a pour rôle de communiquer des informations sur le pack batterie au reste du véhicule, par exemple au tableau de bord pour qu'il affiche une jauge représentant l'énergie restante.

Comme le nombre de cellules est important dans les véhicules électriques actuels, généralement au nombre de 96 dans la série, le BMS est implémenté comme un système distribué. Les cartes de supervision des cellules, en anglais *Cell Sensor Unit (CSU)*, collectent les données et mettent en œuvre l'équilibrage, sous la direction de l'unité de contrôle principale, en anglais *Master Control Unit (MCU)*.

Dans les implémentations actuelles, la communications entre les CSUs et le MCU se fait de manière filaire, par I2C, SPI, ou UART. Dans ce travail, nous avons considéré la possibilité de remplacer ces fils par des communications sans-fils, en nous basant sur les protocoles standardisés de l'Internet des Objets.

En effet, pour des composants embarqués à bas coût, aux faibles capacité de calcul, et disposant de liens de communication bas débit, il n'est pas possible d'embarquer une pile IPv6 et TCP classique. Des protocoles spécialisés ont été développés pour les besoins spécifiques de l'Internet des Objets. Ici, notre choix c'est arrêté sur IEEE Std. 802.15.14-2015 avec le mode TSCH. Dans un réseau d'objets, les nœuds peuvent avoir soit une topologie en étoile, dans laquelle ils communiquent uniquement avec un contrôleur central, soit une topologie maillée, permettant aux paquets d'effectuer de multiples sauts avant d'atteindre leur destination.

Utiliser des communications sans-fil au sein d'un BMS n'est pas un choix intuitif, car les fils de cuivre sont faciles à gérer en terme d'approvisionnement, et leur coût est très bas. Néanmoins, certains avantages peuvent être envisagés quant à l'utilisation de liens sans-fil. Cela simplifie la procédure d'assemblage, et économise un peu de

poids et d'espace au sein du pack batterie. Avoir un microcontrôleur, dont le rôle principal est de gérer les communications, sur chaque CSU ouvre des opportunités en terme de répartition des fonctionnalités logicielles, et grâce à cela l'information pourrait être stockée localement, et réutilisée plus tard, notamment si les éléments du pack devaient être reconfigurés dans une nouvelles application, par exemple du stockage stationnaire. Cette technique est appelée la seconde vie des batteries [24].

Contributions

Le but de ce travail n'est pas de déterminer si utiliser des communications sans-fil pour entre les sous-systèmes d'un BMS est, ou non, la bonne approche. En revanche, le but est d'en étudier la faisabilité et de déterminer quelle est la manière la plus appropriée de le faire.

ÉTAT DE L'ART Comme beaucoup de technologies de communications sans-fil existent, nous proposons tout d'abord un état de l'art en la matière, mais aussi les principales caractéristiques des communications au sein d'un BMS telles qu'elles sont implémentées aujourd'hui.

PERFORMANCE D'UN RÉSEAU D'OBJETS À L'INTÉRIEUR D'UN PACK BATTERIE DE VÉHICULE ÉLECTRIQUE Pour déterminer l'impact de l'environnent véhicule électrique sur le réseau sans-fil, et en particulier de l'intérieur d'un pack batterie, nous avons effectué de nombreuses expériences. Le but ici est de déterminer ce qui cause les pertes de paquets, et quelle est l'influence de la position des nœuds et de l'enceinte du pack batterie sur le pourcentage de paquets perdus.

APPROCHE PAR PROGRAMMATION LINÉAIRE POUR GÉRER LE RÉSEAU Sur la base des résultats obtenus par expérimentation, nous proposons une stratégie de gestion du réseau qui se base sur la technique de Programmation Linéaire. Les résultats de cette technique sont la topologie souhaitée pour le réseau, ainsi que l'ordonnancement des transmissions.

APPROCHE PAR SIMPLE DESCENTE POUR GÉRER LE RÉSEAU La technique mentionnée précédemment peut être améliorée, en particulier en terme de temps de traitement. Nous proposons alors une méthode de calcul plus appropriée à des systèmes embarqués.

PROTOCOLE RÉSEAU POUR PROPAGER LES DÉCISIONS DU GESTIONNAIRE Dans l'approche que nous avons choisie, un contrôleur central est utilisé, et il est donc nécessaire d'avoir un protocole pour propager ses décisions. Nous proposons donc un protocole permettant de gérer la procédure pour rejoindre le réseau, la mise à jour de la

topologie et de la *slotframe*, et qui inclus quelques mécanismes de réparation.

EXPÉRIENCE DANS L'ENVIRONNEMENT VÉHICULAIRE POUR ÉVALUER LA TECHNIQUE PROPOSÉE Afin de vérifier que le protocole et la technique de gestion proposée fonctionnent, nous effectuons une expérience avec un réseau d'objets dans un pack batterie. Les résultats montrent que le système est efficace, mais aussi qu'il est possible de l'améliorer.

B.2 ÉTAT DE L'ART

À PROPOS DES CSUS DANS LES IMPLÉMENTATIONS FILAIRES En regardant de plus près la *datasheet* de plusieurs circuits intégrés conçus pour être utilisés dans un BMS, ainsi que des implémentations, nous pouvons avoir une idée assez précise du volume de données échangées et de leur périodicité. La détection de la sous-tension ou de la surtension est généralement réglée sur une période de 2 secondes. Les communications filaires génèrent très peu d'*overhead*, avec 6 octets pour une requête classique, et 3 octets sont ajoutés dans la réponse en plus de la charge utile. Donc, demander toutes les valeurs mesurables par un bq76PL455A-Q1 nécessite d'échanger 51 octets. Demander toutes les valeurs pour un pack batterie demande 12.68ms avec ce composant, et échanger une paire requête/réponse prend entre 100 et 500µs pour être échangée. Toujours pour ce composant, mesurer une valeur avec un convertisseur analogique numérique prend environ 10µs.

UN ÉTAT DE L'ART SUR LES TECHNOLOGIES DE COMMUNICATION SANS FIL POUR L'INTERNET DES OBJETS Le Bluetooth Low Energy aurait pu être un choix intéressant, car il c'est une technologie bien éprouvée, peu énergivore, et dotée de fonctionnalités pour améliorer la qualité de service. Hélas, son support limité pour les réseaux maillés nous a conduit à l'écarter dans ce travail.

Le Wi-Fi est une autre technologie bien connue, qui permet un débit de communication élevé entre les nœuds. Mais ça consommation d'énergie élevée est un gros défaut pour les applications embarquées fonctionnant sur batterie, les CSUs étant ici alimentées sur les cellules de la batterie de traction du véhicule qu'elles supervisent.

Wi-Fi Halow est une technologie pour les réseaux en étoile, nécessitant une moyenne portée. Plutôt conçue pour relier un réseau d'objets à l'Internet, elle n'est pas vraiment approprié au sujet qui est traité ici.

WirelessHART est une technologie pour les réseaux industriels courte portée. Elle utilise IEEE Std. 802.15.4-2006 pour ses couches physique et liaison, avec TDMA et un mécanisme de saut de canal. La topologie du réseau est maillée. Elle permet le *blacklisting* des canaux

problématiques sur décision de l'administrateur réseau. Le fait que la couche applicative est HART rend cette technologie peu flexible et donc peu attractive pour l'application visée.

ISA100 est un protocole similaire à WirelessHART, mais qui permet plus de flexibilité, notamment à la couche applicative. Il n'est en revanche pas compatible avec IEEE Std. 802.15.4-2006, et le grand nombre d'options configurables font que des équipements provenant de vendeurs différents ont peu de chances d'être compatibles. Aussi, le fait qu'il est directement axé vers les applications industrielles, le standard spécifiant comment le *backbone* réseau doit être déployé, font qu'il n'est pas vraiment adapté à notre cas de figure.

Wireless IO-Link est une autre technologie sans fil pour les réseaux industriels. Bien qu'efficace, elle a été jugée trop simple pour les besoins de notre application, car elle ne permet que des schémas de trafic requête/réponse, et une topologie en étoile.

Enfin, IEEE Std. 802.15.4-2015, avec le mode TSCH, est la technologie qui a été retenue ici. Elle permet une bonne fiabilité, avec l'utilisation de TSCH, est flexible, permettant notamment les réseaux maillés ainsi que n'importe quel schéma de trafic. Ce qui n'est pas spécifié dans le standard est comment construire la *slotframe*, c'est à dire le schéma qui représente l'ordonnancement des transmissions, et se répète au cours du temps.

TRAVAUX CONNEXES SUR L'UTILISATION DE RÉSEAUX SANS-FIL À L'INTÉRIEUR D'UN PACK BATTERIE DE VÉHICULE ÉLECTRIQUE

D'autres ont déjà commencé à étudier cette possibilité. Alonso *et al.* [76] ont évalué les performances de différentes antennes à l'intérieur d'une enceinte batterie, et recommandent d'utiliser des antennes PIFA dans la bande des 2,4GHz.

La chaîne de traction, et en particulier le moteur du véhicule, émet des rayonnements électromagnétiques, mais qui se trouvent en dessous des 100kHz [75], et ne devraient donc pas interférer avec le réseau sans-fil.

Dans [56], les auteurs étudient un réseau sans fil à bord d'un véhicule, où tous les nœuds sont à portée les uns des autres, et préconisent d'utiliser une topologie en arbre à 2 niveaux (en plus du nœud racine), afin d'utiliser les chemins les plus fiables. Ils suggèrent aussi d'utiliser un mécanisme d'agrégation des données aux nœuds intermédiaires.

B.3 PERFORMANCE D'UN RÉSEAU IOT DANS UN PACK BATTERIE DE VÉHICULE ÉLECTRIQUE

Afin de connaître précisément la qualité moyenne des liens réseau avec la technologie choisie à l'intérieur d'un pack batterie, mais aussi de connaître l'impact de la position et des potentielles interférences

avec d'autres utilisateurs de la bande des 2,4GHz, de nombreux tests ont été réalisés.

Tout d'abord dans une boîte métallique que nous avons construite, similaire à celle présentée dans [76]. Ce test a été peu concluant, car la qualité des liens étaient parfaite.

Ensuite nous avons, en partenariat avec l'école IMT Mines d'Alès, réalisé un test de qualité de lien avec des nœuds déployés à l'intérieur du pack batterie du véhicule prototype PGO e-Hemera. Les performances se sont montrées élevées, avec un pourcentage de paquets délivrés de 99,4% et plus. La topologie était en étoile, et les nœuds présentant le plus de pertes étaient les plus éloignés et les moins en vue du nœud racine, certaines pièces métalliques empêchant une communication directe.

Nous disposons également d'un pack batterie de Renault Fluence à l'école, avec lequel nous avons réalisé un test similaire. Le pourcentage de de paquets délivrés était de 98,7% et plus, montrant encore une fois de bonnes performances de base du réseau à l'intérieur d'un pack batterie.

Cette plate-forme nous a également servie pour effectuer un test d'interférence. Nous avons placé des nœuds réseau utilisant intensivement le Wi-Fi sur les canaux les plus courants (1, 6 et 11), et mesuré l'impact sur le pourcentage de paquets délivrés par canal. Il est clair que sur les canaux IEEE Std. 802.15.4 et Wi-Fi qui se chevauchent, le taux de perte est très important (parfois autour de 50%). Les interférences sont donc un vrai frein aux communications sans fil pour le BMS. Il est en revanche possible d'y remédier en utilisant du *blacklisting* temporaire des canaux problématiques.

Enfin, nous avons utilisé la plate-forme *Open-Source Vehicle (OSV)* disponible à l'école pour effectuer un test d'impact de conduite du véhicule sur le réseau sans-fil. Comme cela était indiqué dans la littérature, les organes électroniques du véhicule ne semblent pas perturber de manière significative le réseau sans-fil. En revanche, les réseaux Wi-Fi avoisinant ont semblé interférer avec le réseau sans-fil sur les canaux qui se chevauchent.

Globalement, tout ces tests montrent qu'il est techniquement possible d'utiliser un réseau sans-fil à l'intérieur d'un pack batterie de véhicule électrique.

B.4 ALGORITHMES DE GESTION DU RÉSEAU

Dans la partie précédente, nous avons effectué des mesures pour évaluer la qualité des liens à l'intérieur de l'enceinte d'un pack batterie. Dans ce chapitre, nous construisons des algorithmes, sur la base de ces résultats, pour construire une topologie en arbre où tous les nœuds sont au maximum à deux sauts de la racine et en utilisant les meilleurs liens, et pour construire une *slotframe* pour TSCH.

Pour mettre au point les deux techniques présentées ci-dessous, nous avons travaillé sur des qualités de liens générées aléatoirement, et de valeur ressemblant les mesures effectuées au chapitre précédent. Nous avons développé une application en C++ pour régler, tester, et évaluer les algorithmes proposés.

TECHNIQUE BASÉE SUR LA PROGRAMMATION LINÉAIRE Ici, nous voulons tout d'abord à déterminer quelle est la meilleure topologie possible, en cherchant à maximiser une fonction objectif représentant la qualité globale des liens.

Nous le faisons à l'aide un programme linéaire en nombre entiers, dans lequel les variables valent 0 pour un lien non utilisé ou 1 pour un lien utilisé, dans lequel les contraintes sont les suivantes :

- Chaque nœud a un nombre maximal d'enfants (en général 2 du fait de la taille des paquets de données et de la taille d'un paquet qu'il est possible de transmettre dans un *timeslot*).
- Chaque nœud a un parent.
- Chaque nœud a un chemin vers la racine.

Pour cette topologie, nous cherchons ensuite à construire une *slotframe*, toujours en utilisant la programmation linéaire. La fonction objectif a une valeur proportionnelle à la taille de la *slotframe*, et ici nous cherchons à la maximiser. Les contraintes du programme linéaire avec des variables binaires, valant 0 quand le *timeslot* n'est pas utilisé, et 1 quand il l'est, sont les suivantes :

- Si un nœud passe par un nœud intermédiaire afin d'envoyer son paquet au nœud racine, alors ce paquet doit être envoyé avant le paquet du nœud intermédiaire dans la *slotframe*.
- Il y a un nombre maximum de canaux utilisables fixé au préalable.
- Chaque nœud ne peut utiliser sa radio qu'une seule fois pour chaque *slot offset*.
- Chaque nœud (excepté le nœud racine) doit avoir au moins un *timeslot* d'émission d'alloué.
- Pour chaque nœud, le taux de paquets délivrés sur le chemin vers la racine doit être supérieur à un seuil constant fixé au préalable.

Nous avons testé cette technique dans divers cas de figure, et constaté qu'elle donne des résultats satisfaisants, notamment du fait que, par rapport à une topologie basique comme la topologie en étoile, la qualité des chemins peut être augmentée de manière significative tout en gardant une taille de *slotframe* raisonnable, malgré les *timeslots* de retransmissions qui sont programmés.

TECHNIQUE BASÉE SUR LA SIMPLE DESCENTE Le problème qui a été identifié avec la technique par programmation linéaire est le temps d'exécution. La méthode par simple descente est connue pour donner une solution approchée (parfois optimale) à un problème d'optimisation, avec un temps d'exécution court et qui peut être borné.

Ici nous ne traitons que la gestion de la topologie, car l'allocation des *timeslots* peut être géré par un algorithme procédural classique, tout en satisfaisant la contrainte sur le taux de paquets délivrés.

La fonction objectif retenue pour calculer la topologie est la même que dans la technique par programmation linéaire, afin de pouvoir comparer facilement les deux méthodes. Cependant, afin de bien évaluer ces deux méthodes, une autre technique a été retenue par la suite : la moyenne des qualités de liens pondérés (par le nombre d'enfants du nœud plus 1).

La technique par simple descente se déroule en deux étapes : tout d'abord une solution valide est construite au regard des contraintes. Puis, dans un deuxième temps, diverses permutations sont essayées sur cette solution, afin de voir laquelle améliore le plus la solution, à chaque itération. Quand plus aucune permutation n'améliore la solution, celle-ci est considérée comme étant la solution finale au problème. La méthode retenue pour construire la solution initiale est la topologie en étoile, et les permutations utilisées sont :

- Échanger un nœud de rang 1 avec un nœud de rang 2.
- Échanger deux nœuds de rang 2.
- Placer un nœud de rang 1 comme enfant d'un autre nœud de rang 1.

ÉVALUATION DES PERFORMANCES ET RÉGLAGES DES PARAMÈTRES Nous avons effectué de nombreux calculs de topologie et de *slotframe* pour un grand nombre de solutions initiales, et avec divers valeurs pour les paramètres de la fonction objectif. Avec cette méthode, nous avons pu trouver les valeurs les plus appropriées pour les paramètres de la fonction objectif. Mais aussi, nous avons observé que la technique par simple descente est effectivement plus rapide à retourner un résultat.

B.5 PROTOCOLE RÉSEAU POUR LA GESTION DE LA BATTERIE

Dans la partie précédente, nous avons conçu et évalué des algorithmes qui permettent de construire une topologie et une *slotframe* dans un cas précis pour notre cas d'usage, en présupposant une connaissance *a priori* des qualités de lien au sein du réseau. Ici, nous utilisons ces algorithmes de manière itérative, en supposant des qualités de lien de 100% pour tous les liens à la formation du réseau, afin d'explorer et

découvrir au fur et à mesure quelles sont vraiment les qualités des différents liens possibles et converger vers la meilleure topologie.

Pour ce faire, nous avons aussi proposé une architecture et un protocole permettant de propager l'information, et en particulier les décisions du gestionnaire de réseau. Bien que la qualité de service soit l'enjeu ici pour le plan de données, ce protocole de gestion du réseau a été conçu comme asynchrone et réactif, basé principalement sur des messages périodiques plutôt que des mécanismes de retransmission.

Dans l'architecture que nous proposons, le nœud racine est au centre, échangeant des messages avec le gestionnaire de réseau d'un côté par une interface filaire, et avec le réseau d'objets de l'autre. Quatre messages types de messages, portés par des paquets ICMPv6 spécifiques définis par notre protocole, permettent aux nœuds de participer au réseau d'objets :

- *BMS Network Advertisement* : permet au nœud racine (et lui seul) d'annoncer le réseau pour la supervision du pack batterie.
- *BMS Network Join Request* : permet à un nœud qui n'est pas la racine de demander au nœud racine de rejoindre le réseau d'objets.
- *BMS Network Topology Update* : permet au nœud racine de demander à un autre nœud de changer son parent dans la topologie, ou les *timeslots* qu'il utilise dans la *slotframe*.
- *BMS Network Topology Update* : permet à un nœud qui n'est pas la racine de communiquer la qualité du lien qu'il a avec son parent au nœud racine, afin que cette information soit transmise au gestionnaire de réseau, pour qu'il prenne des décisions quant à la mise à jour de la topologie et la *slotframe*.

Dans ce système, le gestionnaire de réseau qui reçoit les données de qualité de lien environ toutes les 10 secondes, procède à un re-calcul de la topologie toutes les 20 secondes. Si la nouvelles topologie est meilleure que l'ancienne, il déclenche la mise à jour. Le protocole dispose aussi de quelques mécanismes de réparation du réseau, si un nœud venait à perdre sa connexion pour un certain temps.

Afin d'évaluer ce protocole, nous l'avons implémenté dans Contiki-NG, et déployé sur des nœuds I3Mote CC2650. Nous les avons placés dans le pack batterie de Renault Fluence dont nous disposons, et avons observé le réseau évoluer pendant environ une heure, chaque nœud autre que la racine envoyant un paquet de données par *slotframe*. Le résultat obtenu est proche de ce qui était attendu : la topologie a beaucoup changé, notamment au début de l'expérience, puis a convergé et s'est stabilisée. Une fois que la dernière topologie a été décidée, les qualités des liens entre les nœuds de rang 1 et la racine sont bien plus élevées que la moyenne de ce qui avait été mesuré dans

la partie évaluation, avec une valeur de 99,8% de paquets délivrés et plus, et la topologie est restée stable pendant 6 minutes, toujours avec une réévaluation toutes les 20 secondes. Des points d'amélioration sont possibles pour ce système, notamment au moment des pertes de paquets ont été constatées au moment des changements de topologie. Utiliser des mécanismes non destructifs pour allouer les nouveaux *timeslots*, et conserver les anciens pour une certaine durée avant de les oublier (par exemple pendant une minute), serait un moyen de palier à ce problème.

B.6 CONCLUSION

Le BMS d'un pack batterie est un composant absolument nécessaire d'un pack batterie pour garantir sa sûreté de fonctionnement. Ce travail a montré qu'il est techniquement possible d'utiliser des communications sans-fil entre les sous-systèmes d'un BMS.

Cependant, déployer un tel système ne peut être fait sans prendre en considération les aspects cyber-sécurité. En effet, avec des communications sans-fil le medium est partagé, ce qui rend possible, dans une certaine mesure, les attaques par brouillage.

Aussi, un certain nombre de dispositions doivent être prises dans les couches plus hautes pour garantir la fiabilité des données à la couche applicative, telles que l'authentification, l'intégrité, la traçabilité, la confidentialité, et la disponibilité.

Par ailleurs, la gestion de l'endormissement et du réveil du BMS utilisant des communications sans-fil, et de ses sous-systèmes, est un enjeu majeur pour la viabilité de cette approche. Utiliser une couche liaison spécifique pour les périodes où le véhicule est dans un mode de sommeil est probablement une bonne approche pour gérer ce défi.

Ensuite, la seconde vie des batteries est à prendre en considération dans le développement de ce système électronique et informatique. Les communications sans-fil peuvent être bénéfiques pour ce cas d'usage, sous réserve que les interfaces des CSUs soient uniformisées, permettant l'interopérabilité avec le MCU utilisé dans la nouvelle application.

L'attrait pour le BMS avec des communications sans-fil dépend de plusieurs aspects. L'argument concernant la réduction de la masse du pack batterie semble assez faible, car supprimer les fils permettraient de gagner une vingtaine de grammes sur un pack de plus de 200 kilogrammes.

Enfin, la disponibilité des composants, et la complexité induite, sont aussi des obstacles à l'adoption de communications sans-fil au sein du BMS. En résumé, le BMS utilisant des communication sans-fil est techniquement réalisable et peut être fiable. Il présente des avantages, mais aussi des inconvénients, et les acteurs de l'industrie automobile devront trancher quant à son avenir.

PUBLICATIONS

[1] G. Le Gall, N. Montavont, and G. Z. Papadopoulos, "Enabling IEEE 802.15.4-2015 TSCH based Wireless Network for Electric Vehicle Battery Management," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, 2020, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ISCC50000.2020.9219551

[2] G. Le Gall, N. Montavont, and G. Z. Papadopoulos, "Iot network management within the electric vehicle battery management system," *Journal of Signal Processing Systems*, Jul 2021. [Online]. Available: https://doi.org/10.1007/s11265-021-01670-2

[3] C. Alderete, G. Le Gall, A. Marquet, G. Z. Papadopoulos, and N. Montavont, "Wireless network for in-car communication," in *Ad-hoc, Mobile, and Wireless Networks*, N. Montavont and G. Z. Papadopoulos, Eds. Cham: Springer International Publishing, 2018, pp. 138–144. [Online]. Available: https://doi.org/10.1007/978-3-030-00247-3_14

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

6LowPAN IPv6 over Low-Power Wireless Personal Area Networks. 9, 22, 29, 77
6P 6top protocol. 25

ACK Acknowledgement. 76, 82
ADC Analog to Digital Converter. 6, 11, 16
AODV Ad hoc On-Demand Distance Vector. 20
API Application Programming Interface. 10, 21
ASIC Application Specific Integrated Circuit. 6, 7, 10, 15, 18, 109
ASN Absolute Slot Number. 24

BLE Bluetooth Low Energy. 19, 29, 35, 37, 42, 55, 57
BMS Battery Management System. 4–8, 10–13, 15, 16, 18, 25–27, 29, 34, 37, 42, 50, 55, 58, 60, 65, 75, 77, 78, 80, 81, 87, 88, 96, 97, 99–103, 105–107, 109–111, 113–116, 118, 121, 122, 124, 128, 129
BNA BMS Network Advertisement. 76, 79–81, 84, 86, 87, 91, 96, 97
BNJR BMS Network Join Request. 80, 86, 87, 91
BNTI BMS Network Topology Information. 80–82, 87, 91
BNTU BMS Network Topology Update. 77, 80, 81, 83, 84, 86–91, 95, 96, 128, 130

CAN Controller Area Network. 4, 6, 25, 26, 29, 102, 113
CoAP Constrained Application Protocol. 10, 12, 36, 77
CPU Central Processing Unit. 11, 70
CRC Cyclic Redundancy Check. 16, 17
CSMA Carrier Sense Multiple Access. 23, 24, 29, 77
CSMA-CA Carrier Sense Multiple Access with Collision Avoidance. 20
CSU Cell Sensor Unit. 6–8, 10–12, 15–18, 26, 29, 60, 62, 87, 99, 102–107, 114–116, 122, 124

DIO Destination Information Object. 76, 79, 87
DODAG Destination Oriented Direct Acyclic Graph. 28
DoS Denial of Service. 100

EB Enhanced Beacon. 87, 91
EDCA Enhanced Distributed Channel Access. 20
EMI Electromagnetic Interference. 27
EV Electric Vehicle. 1, 7, 12, 13, 35, 37, 50, 51, 55, 59, 98, 99, 103, 105, 129

GFSK Gaussian Frequency Shift Keying. 22
GLPK Gnu Linear Programming Kit. 60

HART Highway Addressable Remote Transducer. 21, 22
HTTP Hypertext Transfer Protocol. 10

I2C Inter-Integrated Circuit. 15, 114
IC Integrated Circuit. 5, 16–18
ICMPv6 Internet Control Message Protocol for IPv6. 77, 79, 121
IDE Integrated Development Environment. 31
IETF Internet Engineering Task Force. 25
IoT Internet of Things. 8–10, 13, 15, 19, 20, 25, 34, 35, 37, 46, 47, 77, 78, 124
IPv6 Internet Protocol version 6. 8, 9, 19, 20, 22, 27, 76, 78–82, 84, 114
ISA International Society of Automation. 21
ISM Idustrial, Scientific and Medical. 19, 21, 37

LDO Low-dropout regulator. 18
LED Light-emitting Diode. 111
LP Linear Programming. 13, 59, 60, 66, 69–73, 99, 127

MAC Medium Access Control. 12, 19–21, 34, 36–51, 53–57, 62, 70, 76, 77, 81–83, 87, 91, 101–103, 125–127, 129
MCU Master Control Unit. 6, 7, 15, 16, 78, 96, 99–102, 104–107, 109, 112, 114, 122
MOSFET Metal Oxide Semiconductor Field Effect Transistor. 7
MPLS Multi-Protocol Label Switching. 28

NMI Network Manager Information. 85, 87
NMU Network Manager Update. 78, 80, 87, 89, 96

OCV Open-Circuit Voltage. 2, 4, 112
OF Objective Function. 9, 59, 61–63, 66, 69, 76
OS Operating System. 32, 75
OSI Open Systems Interconnection. 10, 124
OSV Open-Source Vehicle. 50–55, 57, 118, 126

PAN Personal Area Network. 9, 23, 24, 86, 87, 124
PC Personal Computer. 30, 32, 36, 78, 112
PCB Printed Circuit Board. 109
PDF Portable Document Format. 30, 31
PDR Packet Delivery Ratio. 13, 36–51, 53–57, 60, 61, 63–65, 69–73, 76, 83, 85, 91–94, 96, 97, 99, 125–129
PER Packet Error Rate. 65, 91, 100
PHY Physical. 19, 38–40, 42, 46, 50, 51, 55, 77
PIFA Planar Inverted-F Antenna. 26, 35
PLC Power Line Communication. 107

QoS Quality of Service. 10, 19, 35, 75, 98, 99

REST Representational State Transfer. 10
RM Radio Metric. 20

RPL Routing Protocol for Low-Power and Lossy Networks. 9, 19, 27–29, 36, 75–79, 87

SD Simple Descent. 59, 66, 70–73, 91, 99, 100, 107, 127
SOC State of Charge. 2–6, 99, 102, 112, 124
SOH State of Health. 4, 103
SPI Serial Peripheral Interface. 6, 15, 16, 114
SWG Standard Wire Gauge. 104

TCP Transmission Control Protocol. 8, 9, 114
TDMA Time division multiple access. 21, 27, 116
TI Texas Instrument. 35
TSCH Time Slotted Channel Hopping. 9, 11, 12, 15, 23–25, 28, 29, 33–35, 55, 58, 60, 62, 63, 77, 79, 99, 101, 102, 114, 117, 118, 124

UART Universal Asynchronous Receiver Transmitter. 6, 15–17, 114
UDP User Datagram Protocol. 9, 10, 12, 22, 29, 36, 42, 55, 77, 82
USB Universal Serial Bus. 36, 79, 82, 124

WLAN Wireless Local Area Networks. 19
WSN Wireless Sensor Networks. 8, 9, 12, 13, 15, 27, 59, 97, 101, 107
WuR Wake-Up Radio. 102, 103

BIBLIOGRAPHY

[4] B. Multon, "Renewable energy and electric vehicles," ENS Rennes, Tech. Rep., 2018. [Online]. Available: https://github.com/VGuichon/Hackathon-CampOSV-mars-2018/blob/master/B%20Multon%20Conf%2015-3-2018%20CampOSV.pdf

[5] H. J. Bergveld, W. S. Kruijt, and P. H. L. Notten, *Battery Management Systems*. Dordrecht: Springer Netherlands, 2002, pp. 9–30.

[6] C. Shepherd, "Theoretical design of primary and secondary cells. part 3. battery discharge equation," NAVAL RESEARCH LAB WASHINGTON DC, Tech. Rep., 1963.

[7] X. Hu, S. Li, H. Peng, and F. Sun, "Robustness analysis of State-of-Charge estimation methods for two types of Li-ion batteries," *Journal of Power Sources*, vol. 217, pp. 209 – 219, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378775312009937

[8] "Renault Zoé – Véhicule électrique de série – Guide pour les Services de Secours," Renault, Tech. Rep., 2012. [Online]. Available: http://iuv.sdis86.net/wp-content/uploads/2015/07/RENAULT_ZOE_2012.pdf

[9] C. Weng, J. Sun, and H. Peng, "A unified open-circuit-voltage model of lithium-ion batteries for state-of-charge estimation and state-of-health monitoring," *Journal of Power Sources*, vol. 258, pp. 228 – 237, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378775314002092

[10] J. Blackman and S. Monroe, "Overview of 3.3V CAN (Controller Area Network) Transceivers," Texas Instruments, Tech. Rep., Jan. 2013, http://www.ti.com/lit/an/slla337/slla337.pdf.

[11] H. A. Dang, C. Guyon, B. Delinchant, P. Béguery, and F. Wurtz, "Gestion de l'énergie électrique dans l'habitat, cas du stockage électrochimique," in *AUCG-IBPSA 2012*, 2012.

[12] "Battery Chargers and Charging Methods," https://www.mpoweruk.com/chargers.htm, accessed: 2018-06-21.

[13] W.-Y. Chang, "The State of Charge Estimating Methods for Battery: A Review," *ISRN Applied Mathematics*, vol. 2013, 2013.

[14] P. Weicker, *A Systems Approach to Lithium-Ion Battery Management*. Artech house, 2013.

[15] F. Sun, X. Hu, Y. Zou, and S. Li, "Adaptive Unscented Kalman Filtering for State of Charge Estimation of a Lithium-Ion Battery for Electric Vehicles," *Energy*, vol. 36, no. 5, pp. 3531 – 3540, 2011.

[16] A. Jossen, V. Spath, H. Doring, and J. Garche, "Battery Management Systems (BMS) for Increasing Battery Life Time," in *21st International Telecommunications Energy Conference. INTELEC '99 (Cat. No.99CH37007)*, 1999, pp. 56–.

[17] J. Wang, B. Cao, Q. Chen, and F. Wang, "Combined State of Charge Estimator for Electric Vehicle Battery Pack," *Control Engineering Practice*, vol. 15, no. 12, pp. 1569 – 1576, 2007.

[18] "Multi-Cell Li-Ion Battery Management System Using MSP430F5529 and bq76PL536," Texas Instruments, Tech. Rep., dec 2010. [Online]. Available: http://www.ti.com/analog/docs/litabsmultiplefilelist.tsp?literatureNumber=slaa478&docCategoryId=1&familyId=413#

[19] "LTC6803-1/LTC6803-3 Multicell Battery Stack Monitor," Linear Technology, Tech. Rep., 2011. [Online]. Available: http://www.analog.com/en/products/ltc6803-1.html

[20] "bq2947 Overvoltage Protection for 2-Series to 4-Series Cell Li-Ion Batteries with External Delay Capacitor," Texas Instruments, Tech. Rep., jul 2018. [Online]. Available: http://www.ti.com/lit/ds/symlink/bq2947.pdf

[21] "bq76PL455A-Q1 16-Cell EV/HEV Integrated Battery Monitor and Protector," Texas Instruments, Tech. Rep., apr 2015. [Online]. Available: http://www.ti.com/lit/gpn/bq76pl455a-q1

[22] J. Cao, N. Schofield, and A. Emadi, "Battery Balancing Methods: A Comprehensive Review," in *Vehicle Power and Propulsion Conference, 2008. VPPC'08. IEEE*. IEEE, 2008, pp. 1–6.

[23] J. Gallardo-Lozano, E. Romero-Cadaval, M. I. Milanes-Montero, and M. A. Guerrero-Martinez, "Battery equalization active methods," *Journal of Power Sources*, vol. 246, pp. 934–949, 2014.

[24] E. Martinez-Laserna, E. Sarasketa-Zabala, I. Villarreal Sarria, D. Stroe, M. Swierczynski, A. Warnecke, J. Timmermans, S. Goutam, N. Omar, and P. Rodriguez, "Technical Viability of Battery Second Life: A Study From the Ageing Perspective," *IEEE Transactions on Industry Applications*, vol. 54, no. 3, pp. 2703–2713, May 2018.

[25] "bq769x0 3-series to 15-series cell battery monitor family for li-ion and phosphate applications," Texas Instrument, Tech. Rep., 2019. [Online]. Available: https://www.ti.com/lit/ds/symlink/bq76940.pdf

[26] "IEEE Standard for Low-Rate Wireless Networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, April 2016.

[27] R. Alexander, A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, and T. Winter, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, Mar. 2012.

[28] G. Montenegro, N. Kushalnagar, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," IETF RFC 4944, Sep. 2007.

[29] J. Postel, "User Datagram Protocol," RFC 768, Aug. 1980. [Online]. Available: https://rfc-editor.org/rfc/rfc768.txt

[30] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, Jun. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7252.txt

[31] K. Pister, T. Phinney, P. Thubert, and S. Dwars, "Industrial Routing Requirements in Low-Power and Lossy Networks," RFC 5673, Oct. 2009.

[32] X. Vilajosana, K. Pister, and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration," RFC 8180, May 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8180.txt

[33] Q. Wang, X. Vilajosana, and T. Watteyne, "6TiSCH Operation Sublayer Protocol (6P)," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-6tisch-6top-protocol-12, June 2018, http://www.ietf.org/internet-drafts/draft-ietf-6tisch-6top-protocol-12.txt. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-6tisch-6top-protocol-12.txt

[34] Bluetooth SIG, "Bluetooth Core Specification v5.0," Bluetooth SIG, Tech. Rep., Dec. 2016, https://www.bluetooth.com/specifications/bluetooth-core-specification.

[35] Bluetooth Mesh Working Group, "Bluetooth Specification — Bluetooth Mesh Profile," Bluetooth SIG, Tech. Rep., Jul. 2017, https://www.bluetooth.com/specifications/mesh-specifications.

[36] R. T. Yazicigil, P. Nadeau, D. Richman, C. Juvekar, K. Vaidya, and A. P. Chandrakasan, "Ultra-Fast Bit-Level Frequency-Hopping Transmitter for Securing Low-Power Wireless Devices," in *2018*

IEEE Radio Frequency Integrated Circuits Symposium (RFIC), 2018, pp. 176–179.

[37] J. Lindh, C. Lee, and M. Hernes, "Measuring Bluetooth Low Energy Power Consumption," Texas Instruments, Tech. Rep., Jan. 2017, http://www.ti.com/lit/an/swra478c/swra478c.pdf.

[38] H. S. Kim, J. Lee, and J. W. Jang, "BLEmesh: A Wireless Mesh Network Protocol for Bluetooth Low Energy Devices," in *2015 3rd International Conference on Future Internet of Things and Cloud*, Aug 2015, pp. 558–563.

[39] J. Nieminen, T. Savolainen, M. Isomaki, B. Patil, Z. Shelby, and C. Gomez, "IPv6 over BLUETOOTH(R) Low Energy," RFC 7668, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7668.txt

[40] "IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pp. 1–3534, Dec 2016.

[41] W. S. Conner, J. Kruys, K. J. Kim, and J. C. Zuniga, "IEEE 802.11s Tutorial — Overview of the Amendment for Wireless Local Area Mesh Networking," in *IEEE 802 Plenary, Dallas*. IEEE, November 2006. [Online]. Available: http://www.ieee802.org/802_tutorials/06-November/802.11s_Tutorial_r5.pdf

[42] S. R. Das, C. E. Perkins, and E. M. Belding-Royer, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561, Jul. 2003. [Online]. Available: https://rfc-editor.org/rfc/rfc3561.txt

[43] K. Gomez, T. Rasheed, R. Riggio, D. Miorandi, C. Sengul, and N. Bayer, "Achilles and the Tortoise: Power Consumption in IEEE 802.11n and IEEE 802.11g Networks," in *2013 IEEE Online Conference on Green Communications (OnlineGreenComm)*, Oct 2013, pp. 20–26.

[44] L. Qiao, Z. Zheng, W. Cui, and L. Wang, "A Survey on Wi-Fi HaLow Technology for Internet of Things," in *2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2)*, 2018, pp. 1–5.

[45] E. Khorov, A. Lyakhov, I. Nasedkin, and R. Yusupov, "Poster: Fast and Reliable Alert Delivery in Wi-Fi HaLow Sensor Networks," in *2019 IFIP Networking Conference (IFIP Networking)*, 2019, pp. 1–2.

[46] M. S. Meera and S. N. Rao, "A Survey of the State of the Art of 802.11ah," in *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, 2017, pp. 1–4.

[47] B. Lydon, "ISA100 Wireless Standard Review Control & Instrument Manufacturer Perspective," Automation.com, 2011.

[48] M. Nixon and T. R. Rock, "A Comparison of WirelessHART and ISA100. 11a," *Whitepaper, Emerson Process Management*, pp. 1–36, 2012.

[49] S. Petersen and S. Carlsen, "WirelessHART Versus ISA100.11a: The Format War Hits the Factory Floor," *IEEE Industrial Electronics Magazine*, vol. 5, no. 4, pp. 23–34, 2011.

[50] F. P. Rezha and S. Y. Shin, "Performance Analysis of ISA100.11a Under Interference From an IEEE 802.11b Wireless Network," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 919–927, 2014.

[51] J. Shi, M. Sha, and Z. Yang, "Distributed Graph Routing and Scheduling for Industrial Wireless Sensor-Actuator Networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1669–1682, 2019.

[52] "HART Technology Detail," <https://fieldcommgroup.org/technologies/hart/hart-technology-detail>, 2020, Accessed: 2020-07-28.

[53] O. Khader and A. Willig, "An energy consumption analysis of the Wireless HART TDMA protocol," *Comput. Commun.*, vol. 36, pp. 804–816, 2013.

[54] ISA100 Wireless Compliance Institute, "ISA100 WCI Webinar – End to end SIL2 wireless," Tech. Rep., 2019. [Online]. Available: https://isa100owci.org/en-US/Documents/PDF/End-to-end-SIL2-wireless-ISA100.aspx

[55] IO-Link Community, "IO-Link Wireless System Extensions," Tech. Rep., Sep. 2017. [Online]. Available: https://io-link.com/share/Downloads/System-Extensions/IO-Link_Wireless_10112_d095_Sep17.pdf

[56] R. Tavakoli, M. Nabi, T. Basten, and K. Goossens, "Topology Management and TSCH Scheduling for Low-Latency Convergecast in In-Vehicle WSNs," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2018.

[57] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *Proceedings of the 13th ACM conference on embedded networked sensor systems*. ACM, 2015, pp. 337–350.

[58] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic Aware Scheduling Algorithm for Reliable Low-Power Multi-Hop IEEE 802.15.4e Networks," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*. IEEE, 2012, pp. 327–332.

[59] N. Accettura, M. R. Palattella, G. Boggia, L. A. Grieco, and M. Dohler, "DeTAS: a Decentralized Traffic Aware Scheduling technique enabling IoT-compliant Multi-hop Low-power and Lossy Networks," in *Second IEEE WoWMoM Workshop on the Internet of Things: Smart Objects and Services, IoT-SoS*, vol. 26, 2013.

[60] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia, and M. Dohler, "Decentralized Traffic Aware Scheduling in 6TiSCH Networks: Design and Experimental Evaluation," *IEEE Internet of Things Journal*, 2015.

[61] V. Kotsiou, G. Papadopoulos, P. Chatzimisios, and F. Theoleyre, "Is Local Blacklisting Relevant in Slow Channel Hopping Low-Power Wireless Networks?" in *ICC 2017 : IEEE International Conference on Communications*, Paris, France, May 2017, pp. 1 – 7. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01656125

[62] A. Morell, X. Vilajosana, J. L. Vicario, and T. Watteyne, "Label switching over IEEE802. 15.4 e networks," *Transactions on Emerging Telecommunications Technologies*, vol. 24, no. 5, pp. 458–475, 2013.

[63] G. Daneels, B. Spinnewyn, S. Latré, and J. Famaey, "ReSF: Recurrent Low-Latency Scheduling in IEEE 802.15.4e TSCH networks," *Ad Hoc Networks*, vol. 69, pp. 100 – 114, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870517302019

[64] I. Hosni and F. Théoleyre, "Self-healing distributed scheduling for end-to-end delay optimization in multihop wireless networks with 6tisch," *Computer Communications*, vol. 110, pp. 103 – 119, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366416306156

[65] R. Soua, P. Minet, and E. Livolant, "Wave: a Distributed Scheduling Algorithm for Convergecast in IEEE 802.15.4e Networks (Extended Version)," Inria, Research Report RR-8661, Jan. 2015, wireless Networks. [Online]. Available: https://hal.inria.fr/hal-01100420

[66] T. Chang, T. Watteyne, Q. Wang, and X. Vilajosana, "LLSF: Low Latency Scheduling Function for 6TiSCH Networks," in *2016*

International Conference on Distributed Computing in Sensor Systems (DCOSS), May 2016, pp. 93–95.

[67] R. T. Hermeto, A. Gallais, and F. Theoleyre, "Scheduling for IEEE802.15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: A survey," *Computer Communications*, vol. 114, pp. 84 – 105, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366417301147

[68] J. Sen, "A Survey on Wireless Sensor Network Security," *arXiv preprint arXiv:1011.1529*, 2010.

[69] A. D. Wood and J. A. Stankovic, "Denial of Service in Sensor Networks," *Computer*, vol. 35, no. 10, pp. 54–62, Oct 2002.

[70] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming Sensor Networks: Attack and Defense Strategies," *IEEE Network*, vol. 20, no. 3, pp. 41–47, May 2006.

[71] R. Piyare, A. L. Murphy, C. Kiraly, P. Tosato, and D. Brunelli, "Ultra Low Power Wake-Up Radios: A Hardware and Networking Survey," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2117–2157, Fourthquarter 2017.

[72] N. R. Sabar, L. M. Kieu, E. Chung, T. Tsubota, and P. E. Maciel de Almeida, "A memetic algorithm for real world multi-intersection traffic signal optimisation problems," *Engineering Applications of Artificial Intelligence*, vol. 63, pp. 45–53, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197617300854

[73] M. J. Best and K. Ritter, *Linear Programming Active Set Analysis and Computer Programs*. Prentice Hall Upper Saddle River, NJ, 1985.

[74] D. Alonso, O. Opalko, and K. Dostert, "Physical Layer Performance Analysis of a Wireless Data Transmission Approach for Automotive Lithium-Ion Batteries," in *2015 IEEE Vehicular Networking Conference (VNC)*, Dec 2015, pp. 235–242.

[75] B. Revol, J. Roudet, J.-L. Schanen, and P. Loizelet, "EMI study of three-phase inverter-fed motor drives," *IEEE Transactions on Industry Applications*, vol. 47, no. 1, pp. 223–231, 2011.

[76] D. Alonso, O. Opalko, M. Sigle, and K. Dostert, "Towards a Wireless Battery Management System: Evaluation of Antennas and Radio Channel Measurements Inside a Battery Emulator," in *Proceedings of the 2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, Sept 2014, pp. 1–5.

[77] D. Alonso, O. Opalko, and K. Dostert, "Parametrization of Automotive Lithium-Ion Batteries and Its Influence on the Wireless In-Battery Channel," in *2016 10th European Conference on Antennas and Propagation (EuCAP)*, April 2016, pp. 1–5.

[78] D. Alonso, O. Opalko, and K. Dostert, "Channel Measurements and Simulations with Planar Inverted F-Antennas in an Enhanced Testbed for a Wireless Battery Management System," in *WSA 2015; 19th International ITG Workshop on Smart Antennas*, March 2015, pp. 1–8.

[79] D. Alonso, C. Winkler, O. Opalko, and K. Dostert, "Prototyping of the Physical and MAC Layers of a Wireless Battery Management System," in *2015 IEEE Vehicular Networking Conference (VNC)*, Dec 2015, pp. 267–270.

[80] M. Lee, J. Lee, I. Lee, J. Lee, and A. Chon, "Wireless Battery Management System," in *Proceedings of the Proceedings of the 2013 World Electric Vehicle Symposium and Exhibition (EVS27)*. IEEE, 2013, pp. 1–5.

[81] G. Zimmer, "Wireless Battery Management Systems Highlight Industry's Drive for Higher Reliability," *Linear Technology*, February 2017.

[82] T. Instruments, "CC2650 SimpleLinkTM Multistandard Wireless MCU," Texas Instruments, Tech. Rep., jul 2016. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2650.pdf

[83] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – a systematic literature review," *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009, special Section - Most Cited Articles in 2002 and Regular Research Papers. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584908001390

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Réseau sans-fil pour une gestion fiable de la batterie du véhicule électrique

Mots-clés : Véhicule Électrique, Système de Gestion de Batterie, Internet des Objets, IEEE Std. 802.15.4-2015 TSCH, Programmation Linéaire, Simple Descente.

Résumé : La batterie de traction du véhicule électrique est un composant clé. C'est aussi un système sensible dont la tension et la température des cellules qui le composent doivent être maintenues dans des plages de fonctionnement bien définies. Garantir cela est le rôle du *Battery Management System* (BMS). Le BMS est composé de sous-systèmes, appelés *Cells Sensor Units* (CSU), qui supervisent les cellules et rapportent leur état à un composant central, le *Master Control Unit* (MCU). Dans les implémentations de BMS actuelles, cette communication périodique est effectuée de manière filaire. Dans ce travail nous avons étudié la possibilité de remplacer ce réseau de communication par un réseau sans-fil, en utilisant les protocoles standardisés de l'Internet des Objets.

Nous avons évalué les divers protocoles de communication disponibles, et avons choisi de baser nos travaux sur IEEE 802.15.4-2015 *Time Slotted Channel Hopping* (TSCH). Nous avons ensuite cherché à déterminer quelles sont les stratégies de gestion de la topologie et d'ordonnancement des transmission qui sont les plus adaptées à un tel scénario. Nous avons proposé deux algorithmes pour une gestion centralisée du réseau, basés sur les techniques de Programmation Linéaire et Simple Descente, afin d'optimiser la topologie et la *slotframe*. Aussi, nous avons proposé un protocole de routage qui permet de propager les décisions du gestionnaire de réseau aux nœuds. Enfin, nous avons testé cette solution avec un réseau d'objets dans un environnement véhiculaire.

Title : Wireless Network for Reliable Electric Vehicle Battery Management

Keywords : Electric Vehicles, Battery Management System, Internet of Things, IEEE Std. 802.15.4-2015 TSCH, Linear Programming, Simple Descent.

Abstract : The traction battery of an electric vehicle is a key component. It is also a sensitive system for which the voltage and temperature of the cells it is made of must be kept in a given working range. This is the role of the Battery Management System (BMS). The BMS is made of subsystems, called Cells Sensor Units (CSU), which supervises the cells and report their state to a central component named Master Control Unit (MCU). In current BMS implementations, this periodic communication is performed through wires. In this work, we have studied the possibility to replace this communication network with a wireless network, using standardized protocols of the Internet of Things.

We have evaluated the different communication protocols available, and have chosen to base our work on IEEE 802.15.4-2015 Time Slotted Channel Hopping (TSCH). We then have sought to determine what the most adapted topology and scheduling management strategies for such a scenario are. We have proposed two algorithms for centralized network management, based on the Linear Programming and Simple Descent techniques, in order to optimize the topology and slotframe. Also, we have proposed a routing protocol which allows to propagate the decisions of the network manager to the nodes. Finally, we have tested this solution with a network of objects in a vehicular environment.