FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Mesh Network of Surveillance Cameras Using FM Radio as a Control Channel

João Pedro dos Santos Ribeiro Dias

MASTER IN ELECTRICAL AND COMPUTERS ENGINEERING

Supervisor: Manuel Pereira Ricardo (PhD)

Second Supervisor: Rui Campos (PhD)

Third Supervisor: Filipe Sousa (MSc)

July 24, 2015

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES **2014/2015**

A Dissertação intitulada

"Mesh Network of Surveillance Cameras Using FM Radio as a Control Channel"

foi aprovada em provas realizadas em 16-07-2015

o júri

[illegible]

Presidente Professor Doutor Mário Jorge Moreira Leitão
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

[illegible]

Professor Doutor Adriano Jorge Cardoso Moreira
Professor Associado do Departamento de Sistemas de Informação da Escola de Engenharia da Universidade do Minho

[illegible]

Professor Doutor Manuel Alberto Pereira Ricardo
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

[illegible]

Autor - João Pedro dos Santos Ribeiro Dias

Faculdade de Engenharia da Universidade do Porto

Resumo

Com o crescimento da *Internet of Things* (IoT), o mercado das câmaras IP e dos sistemas de videovigilância baseados nesta tecnologia tem vindo a crescer exponencialmente. As redes de sensores de vídeo sem fios (WVSN), baseadas nas redes emalhadas IEEE 802.11 (Wi-Fi) constituem uma solução adequada e largamente utilizada na construção destes sistemas de videovigilância. No entanto, apesar da sua flexibilidade, alcance, facilidade de instalação e baixo custo, estas redes de sensores de vídeo sem fios apresentam três grandes problemas: mau desempenho nas transmissões, falta de justiça no acesso ao meio (os nós mais próximos da gateway monopolizam o meio) e grandes consumos de energia. Enquanto os primeiros dois problemas comprometem a escalabilidade da rede, a fraca eficiência energética reduz o tempo de vida do sistema quando este é alimentado por baterias e contraria a actual tendência de reduzir a pegada ecológica da Internet.

A maioria das soluções encontradas no estado da arte para resolver os problemas previamente mencionados são propostas no contexto geral das redes de sensores sem fios (WSN) e não têm em conta os requisitos específicos e mais exigentes das redes especializadas na recolha e transmissão de vídeo (WVSN). Além disso, também não existem soluções únicas para os três problemas em simultâneo. Normalmente cada solução foca-se apenas na resolução de um dos problemas em particular.

O objectivo desta tese é por isso desenvolver e implementar uma solução completa e única que possibilite obter uma rede de sensores de vídeo sem fios escalável, com melhor desempenho, justiça no acesso ao meio e energeticamente eficiente. Esta solução consiste num mecanismo de escalonamento das transmissões, controlado por um canal *out-of-band*, baseado no FM Radio Data System (RDS), que permite aos nós desligarem a sua carta Wi-Fi quando não estão a ser utilizados para transmitir, receber ou reencaminhar dados.

Os resultados obtidos no final deste trabalho confirmam que o nosso objectivo foi de facto alcançado. Com a solução implementada, conseguimos reduzir até 48 % o consumo associado ao Wi-Fi, numa rede com 7 nós, mantendo um bom desempenho nas transmissões e um acesso ao meio justo para todos os nós da rede. Desta forma, chegamos à solução única que pretendíamos para resolver os três problemas principais das redes de sensores de vídeo sem fios em simultâneo. Além disso, a rede construída com a nossa solução não só é escalável como também permite aumentar os ganhos energéticos com o aumento do número de nós.

Abstract

Following the rise of the Internet of Things (IoT), the market of IP-based video surveillance systems has been growing exponentially. Wireless Video Sensor Networks (WVSNs), based on IEEE 802.11 (Wi-Fi) mesh networks, are a suitable and widely employed solution to build such systems. Despite their flexibility, coverage area, easiness of installation and low cost, WVSNs suffer from three major problems: poor performance in transmissions, throughput unfairness and energy inefficiency. While the first two compromises the scalability of the system, the energy inefficiency, besides reducing the system lifetime when the nodes are battery powered, also thwarts the current trend of green networks, as well as the attempt of reducing the ever-increasing Internet's carbon footprint.

The majority of the solutions found on the state of the art to address the above-mentioned problems are proposed for the general context of Wireless Sensor Networks (WSNs), not being suitable for the particular and more demanding case of WVSNs. Moreover, few solutions combine approaches to tackle all these problems at once; normally they only focus on a single problem at a time.

Therefore, the goal of this thesis is to develop and implement an all-around solution in order to obtain a fully functional prototype of a WVSN with better performance, throughput fairness, energy-efficiency and scalability. This solution consists of a scheduling mechanism that uses the radio data system (RDS), in the FM band, as an always-on, point-to-multipoint control channel to turn off the nodes' Wi-Fi radio interfaces whenever they are not needed to transmit, receive or relay data.

The results obtained with the proof-of-concept prototype confirm that our goal was effectively reached. With the implemented solution, we were able to achieve gains of up to 48 % in the power consumption associated to Wi-Fi, in a network with 7 nodes, while keeping good performances and throughput fairness, thus achieving the desired solution that addresses the three main problems of WVSNs at once. Moreover, the network achieved would not only be scalable but the energy gains would also increase with the network's growth.

Acknowledgments

I would like to express my gratitude to Prof Manuel Ricardo, my supervisor in this thesis, for allowing me to join this interesting and challenging project and to my other two supervisors, Rui Campos and Filipe Sousa, for their permanent availability to discuss my work and their valuable suggestions to improve it. I would also like to thank to Filipe Ribeiro for his valuable help in the understanding of the PACE implementation, which was the starting point of my work, and for the support provided during my thesis. Filipe Teixeira, Mário Lopes and José Oliveira, other members from our Wireless Networks (WiN) group, have also provided some help during this work by providing components for the testbed, giving suggestions and sporadically helping to solve some problems.

I would also like to thank to my friends and colleagues for their support and for the leisure times with them during this semester, which helped me to recover from the hard work required in the development of this thesis.

I'm grateful to my family too. Besides supporting and sustaining me, they have also helped me in my daily life, allowing me to be fully focused on my thesis during this semester.

Finally, I would like to thank the BEST CASE project ("NORTE-07-0124-FEDER-000056", "NORTE-07-0124-FEDER-000058" and "NORTE-07-0124-FEDER-000060") financed by the North Portugal Regional Operational Programme (ON.2 – O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds, through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT).

João Dias

“Quem quis, sempre pôde.”

Luís Vaz de Camões

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AP Access Point

ARP Address Resolution Protocol

ASIC Application-specific Integrated Circuit

ATCM Active Topology Creation and Maintenance

ATIM Announcement Traffic Indication Message

BLE Bluetooth Low Energy

CPU Central Processing Unit

CRC Cyclic Redundancy Check

CSI Camera Serial Interface

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

DCF Distributed Coordination Function

DGR Directional Geographical Routing

DMA Direct Memory Access

DSP Digital Signal Processor

DVB-T Digital Video Broadcasting — Terrestrial

ESSID Extended Service Set Identification

FEC Forwarding Error Correction

GPIO General Purpose Input Output

GPU Graphics Processing Unit

IBSS Independent Basic Service Set

I^{2}C Inter-Integrated Circuit

IoT Internet of Things

IP Internet Protocol

IPC Inter-process Communication

I^{2}S Integrated Interchip Sound

IT Information Technology

MAC Medium Access Control

MAP Mesh Access Point

MIMO Multiple Input, Multiple Output

MLME MAC Sublayer Management Entity

NIC Network Interface Controller

NTP Network Time Protocol

OEDSR Optimized energy-delay sub-network routing

OSI Open Systems Interconnection

PSM Power Saving Mode

PWM Pulse-Width Modulation

QoS Quality of Service

RDS Radio Data System

RTS/CTS Request-to-send/Clear-to-send

RTSP Real Time Streaming Protocol

SIMD Single-Instruction Multiple-Data

SDR Software Defined Radio

SDRAM Synchronous Dynamic Random-Access Memory

SoC System on a Chip

SPI Serial Peripheral Interface Bus

TCP Transmission Control Protocol

TDMA Time Division Multiple Access

UART Universal Asynchronous Receiver/Transmitter

UDP User Datagram Protocol

USB Universal Serial Bus

VHF Very High Frequency

VLAN Virtual Local Area Network

WMN Wireless Mesh Network

WMSN Wireless Multimedia Sensor Network

WSN	Wireless Sensor Network
WVSN	Wireless Video Sensor Network

Chapter 1

Introduction

1.1 Context

Wireless Sensor Networks (WSNs) were one of the most promising technologies on the past decade. Based on a collaborative effort between low-cost and low-power multi-functional sensor nodes, that are small in size and communicate unrestrained in short distances, these WSNs have revolutionized the way of retrieving data from the environment. Their unique features enabled a wide range of new applications in fields such as health, military and security, becoming increasingly part of our lives [1].

Given the success of WSNs, a multitude of researchers have focused their works in further developing these sensor networks. Eventually, the emergence of inexpensive and miniaturized hardware such as cameras and microphones, along with the success and popularization of Wi-Fi [2], paved the way to the development of Wireless Multimedia Sensor Networks (WMSNs). A reference architecture of these WMSNs is presented in Fig. 1.1.

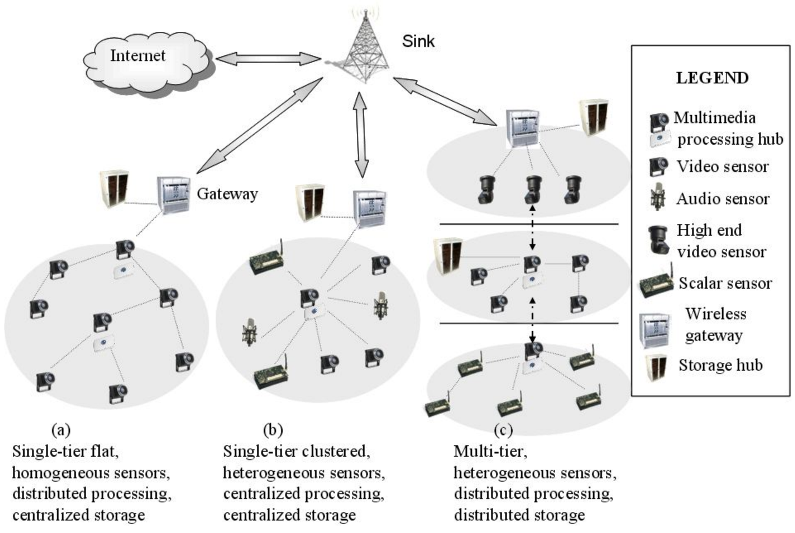

Internet
Sink
Gateway
LEGEND
Multimedia processing hub
Video sensor
Audio sensor
High end video sensor
Scalar sensor
Wireless gateway
Storage hub
(a)
Single-tier flat, homogeneous sensors, distributed processing, centralized storage
(b)
Single-tier clustered, heterogeneous sensors, centralized processing, centralized storage
(c)
Multi-tier, heterogeneous sensors, distributed processing, distributed storage

Figure 1.1: Reference architecture of a wireless multimedia sensor network [3].

WMSNs are fundamentally similar to WSNs. They both rely on large numbers of low-cost and low-power sensor nodes, organized in ad hoc networks and deployed within a certain area, to retrieve data from the environment. The big difference between the two concepts is the type of data retrieved. The goal of a WMSN is to ubiquitously retrieve more descriptive information about the ambience such as video and audio streams, still images, and scalar sensor data, while WSNs are mainly used to detect events or collect measurements [4, 3, 1].This single fact leads to major differences in the design, optimization and operation of WMSNs.

Firstly, the higher amounts of data retrieved require larger bandwidths to be transmitted, thus being impractical the use of the energy-efficient IEEE 802.15.4 technologies, the common option to assure communications in WSNs [5]. Due to its low deployment costs, large bandwidth provided and current popularity, the IEEE 802.11, commonly known as Wi-Fi, is the most promising technology to address this higher bandwidth requirements in WMSNs. However, the Wi-Fi radio interfaces have higher energy consumptions than IEEE 802.15.4 radios, which aggravates the energy efficiency of the network.

Secondly, the real-time applications like video or audio streams have more restrict Quality of Service (QoS) demands. Added to the subsequently need of better and more complex schemes to ensure the QoS requirements, this constraint deteriorates even more the energy-efficiency of the network, which was already impaired due to the use of Wi-Fi. Moreover, in the previous WSNs, it was possible to employ energy-saving mechanisms at the cost of higher transmission delays or lower throughputs because QoS was not critical. Since this cannot be done anymore, the WMSNs, besides consuming more energy, have also less power-saving mechanisms at their disposal.

Despite these extra challenges, WMSNs have also achieved a huge success. Nowadays, with the emergence of the Internet of Things (IoT), WMSNs are becoming more popular than ever.

The most challenging scenario for a WMSN is a system fully composed by video sensors, retrieving video streams and sending them to a sink or a gateway node. This is commonly referred as a Wireless Video Sensor Network (WVSN). A WVSN requires large bandwidths, tight QoS levels and high power consumptions. The high power consumptions increase the deployment costs of such networks, as they usually require the installation of a power line. On the other hand, the large amounts of data generated require an efficient data transmission scheme. Hence, the key to enable the feasibility of such system is by achieving an energy efficient network, with technologies that assure good performances in the video transmissions [6]. These are precisely the challenges that we expect to address in our work, by implementing the FM-WiFIX solution [2].Therefore, throughout this work, we will focus on this kind of networks, although sometimes we use the broader acronym, WMSNs (Wireless Multimedia Sensor Networks).

An obvious application for these WVSNs is a video surveillance system. The video surveillance market is growing fast and keeping up with the emergence and evolution of the IoT (see Fig. 1.2). Thus, IP cameras are quickly overtaking Close-Circuit Television (CCTV) cameras [7]. These WVSNs could add extra value to the standard IP video surveillance system and give an additional boost to this market growth. Therefore, the industry is asking for WVSNs that provide the best image quality and timely transmissions at the lowest energy consumption.

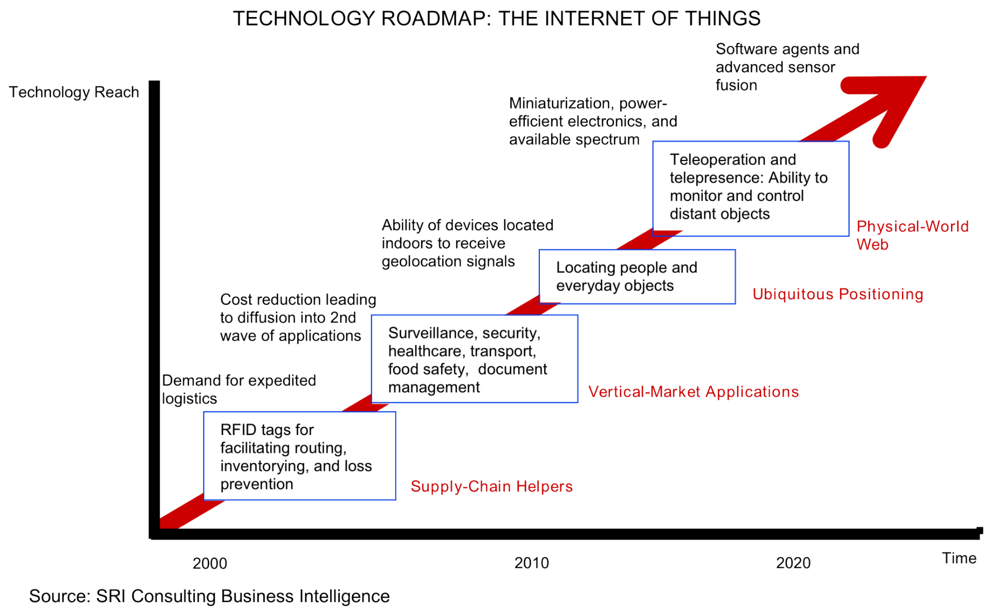

TECHNOLOGY ROADMAP: THE INTERNET OF THINGS
Technology Reach
Software agents and advanced sensor fusion
Miniaturization, power-efficient electronics, and available spectrum
Teleoperation and telepresence: Ability to monitor and control distant objects
Physical-World Web
Ability of devices located indoors to receive geolocation signals
Locating people and everyday objects
Ubiquitous Positioning
Cost reduction leading to diffusion into 2nd wave of applications
Surveillance, security, healthcare, transport, food safety, document management
Vertical-Market Applications
Demand for expedited logistics
RFID tags for facilitating routing, inventorying, and loss prevention
Supply-Chain Helpers
2000
2010
2020
Time
Source: SRI Consulting Business Intelligence

Figure 1.2: The Internet of Things expected evolution [8]. As shown in the graphic, the IoT started connecting surveillance systems to the Internet some years ago but only now the IP cameras have overtaken the CCTV market share.

1.2 Motivation

Within a video monitoring system, an increase in the number of surveillance cameras to monitor a certain location demands the network to be both flexible and capable of supporting numerous flows of video transmitted towards a monitoring center where the video is stored, processed and analyzed. Due to their flexibility, the IEEE 802.11 multi-hop wireless networks are an attractive solution for this problem, allowing an easy and inexpensive setup/removal of cameras; the installation of cameras in inaccessible locations, covering larger geographic areas and the self-organization of the network so that the cameras never lose the connectivity to the monitoring center. The IEEE 802.11 based multi-hop wireless networks offer a satisfactory level of auto-configuration: the nodes operate in ad-hoc mode and share the same frequency channel. Nevertheless, there are three major problems with this kind of networks: poor performance in transmissions, throughput unfairness and energy inefficiency [2]. The first two problems are directly responsible for another problem of these WVSNs: their low scalability.

The performance problem is related to the medium access control mechanism used by Wi-Fi. To control the medium access and avoid frame collisions, IEEE 802.11 uses the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), where a node senses the medium before transmitting. However, in the presence of hidden nodes, this mechanism is not enough to ensure collision-free transmissions. A hidden node scenario is presented in Fig. 1.3a, where a data transmission is taking place from node A to node B. Node C is in the receiver's range (node B) but cannot hear the sender (node A). Thus, if node C wants to transmit data to B at the same time, it will sense the medium and will think that the channel is idle. Node C shouldn't be able to transmit as its transmission will interfere with the data reception at B. However, in this scenario, as node C thinks that the medium is idle, it would transmit at the same time as A, causing a collision in B.

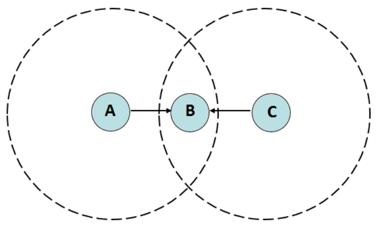
To deal with the hidden node problem, CSMA/CA specifies the Request-to-send/Clear-to-send (RTS/CTS) mechanism which reserves the medium to a specific node that has critical data to transmit, for a given number of slots. By listening a RTS or a CTS message, all neighbor nodes refrain themselves from transmitting packets, avoiding collisions and increasing the system efficiency. Nevertheless, the use of the RTS/CTS mechanism worsens the exposed node problem in ad hoc networks. The exposed node problem is the situation where a given node hears the sender but is out of receiver's range. In this case, the node could transmit to other nodes without causing any collision but refrains itself from transmitting, thinking the medium is busy. A scenario where this problem is aggravated by RTS/CTS is shown in Fig. 1.3b. Node A has data to transmit to B and sends a RTS. After receiving the RTS from A, B transmits a CTS. This CTS is heard by nodes R1-R3, which will then enter in a backoff period. During this period, nodes R1-R3 will not respond to any RTS received, although a simultaneous transmission from a node out of B's range to nodes R1-R3 would not interfere with the transmission from A to B [9, 10].

CSMA/CA and its RTS/CTS mechanism were designed for single-hop networks, operating in the infrastructure mode. Within ad hoc networks, the hidden node problem is much more frequent, due to the multi-hop nature of the network, and the use of RTS/CTS is not advisable as it significant and unnecessarily reduces the network throughput. Actually, it has been shown that the use of RTS/CTS in ad hoc networks may even be counterproductive [10, 11]. Thus, with the increase of hidden nodes, we will have or a great number of collisions or a great number of exposed nodes, if we use the RTS/CTS. Both of these scenarios lead to a reduced network throughput and high end-to-end delays [12, 11]. This is also responsible, in part, for the low scalability, since more nodes in the multi-hop network will mean more hidden/exposed nodes. Improving the transmission performance would allow both an increase in the image quality of the video streams and a more scalable network.

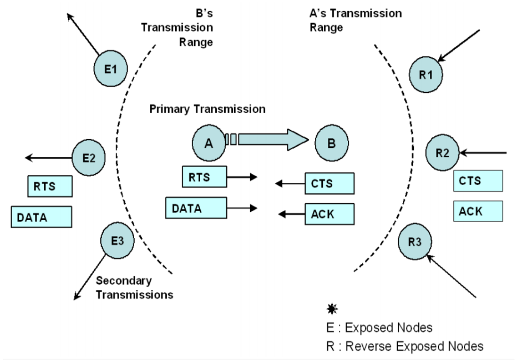
The other cause of the low scalability verified is the throughput unfairness. Due to the multi-hop nature of the network, nodes closer to the gateway monopolize the medium and cause other nodes to starve [12]. Adding new nodes close to the gateway will deteriorate even more the transmission opportunities of farther nodes. On the other hand, adding new nodes away from the gateway means that they will have very few opportunities of transmission, which will probably compromise their operation. A mechanism capable of fairly scheduling the medium access would make the system more scalable and more reliable.

Finally, the energy inefficiency problem is related to the high energy consumption of Wi-Fi interfaces (Table 1.1 shows the power consumption of an IEEE 802.11n interface for the different modes of operation), allied with the high rate of frame collisions and the subsequent retransmission process.

If the video sensor nodes are battery powered, this inefficiency could compromise the system lifetime. However, there is another important motivation to mitigate this problem. According to



(a) The hidden node problem.



(b) The exposed node problem. Here, the RTS from A to B prevents nodes E1-E3 from transmitting, although their transmission wouldn't interfere with A's transmission to B. Similarly, the CTS from B prevents nodes R1-R3 from responding to RTS requests [9].

Figure 1.3: CSMA/CA main problems.

[13], IT-related services now account for 2% of all global carbon emissions. That is roughly as much as the emissions from the global aviation sector. With the emergence of the Internet of Things and the ever-increasing amount of people, devices and services connected (an estimated 2.5 billion people are currently online, and that number is expected to increase by nearly 60% in the next five years [13]), the Internet's carbon footprint is expected to keep growing at a fast pace. Being aware of this serious problem, nowadays, many governments, organizations and companies are showing a growing interest on the "green networking" trend. Thus, any solution that allows reducing the growth of the Internet's energy consumption is welcomed. Back to our context, this means that minimizing the energy consumption of a WVSN is critical, even if the sensors are not battery powered [2].

Mode	**Power Consumption**
IDLE Rx, 1 ANT	0.82 W
IDLE Rx, 2 ANT	1.13 W
IDLE Rx, 3 ANT	1.45 W
Rx SISO, 1ANT	0.94 W
Rx MIMO2, 2 ANT	1.27 W
Rx MIMO3, 3 ANT	1.60 W
Tx SIMO	1.28 W
Tx MIMO2	1.99 W
Tx MIMO3	2.10 W

Table 1.1: IEEE 802.11n power consumption for various Rx and Tx modes [14]. The device used in these measurements was the Intel Wi-Fi Link 5300 a/b/g/n wireless network adapter.

1.3 Objectives

The main goal of this thesis is to implement the FM-WiFIX solution proposed in [2] and achieve a proof-of-concept prototype. To do so, we will build a functional wireless mesh network of surveillance cameras that uses the FM Radio Data System (RDS) as a low power, out-of-band control channel. With this implementation, we expect to improve the performance, the throughput fairness, the scalability and the energy-efficiency of a WVSN used for video monitoring applications, in comparison to the state of the art solutions. After the successful deployment of FM-WiFIX on a real testbed, we also intend to conduct several experiments to evaluate the performance of this solution and to measure metrics such as throughput and energy consumption.

The main milestones that will be considered, in order to successfully accomplish this purpose, are:

- A study about the existing alternatives to the energy-efficient transmission in a real-time video monitoring system (with special relevance regarding those who also try to address the other two major problems: poor performance in transmissions and throughput unfairness) and also about solutions that make use of out-of-band signaling for control purposes;
- The specification and implementation of a control mechanism to schedule the transmissions and the state of the Wi-Fi radio interface, using the RDS control channel;
- Improve the original FM-WiFIX solution to overcome the challenges of a practical implementation.
- Full integration of WiFIX, PACE and FM-WiFIX into a single solution;
- The assembly of a testbed using the Raspberry Pi platform, together with an FM-RDS radio and a Wi-Fi card;
- Build a proof-of-concept prototype of the FM-WiFIX solution.

1.4 Contributions

The main contributions of this thesis are the following:

- Identification and resolution of practical problems in the FM-WiFIX solution;
- Design of a control channel based on the FM Radio Data System (RDS) and respective characterization, regarding its range and error rates;
- Identification of multiple practical problems in the implementation of a low power out-of-band, control channel with such requirements that, to the best of our knowledge, was never tried before;
- Proof-of-concept prototype of the FM-WiFIX solution;
- Characterization of the power consumptions of the hardware used.

1.5 Structure

The rest of this document is organized as follows. Chapter 2 presents the state of the art in solutions that addresses the major problems previously mentioned, with focus on the energy-efficient solutions, suitable for WMSNs and, more specifically, for WVSNs. Chapter 3 describes the FM-WiFIX solution and its companion technologies. In Chapter 4, we describe the modifications done to the original FM-WiFIX solution and all the implementation work done throughout this thesis. The testbed assembled to carry out the evaluations to our system is described in Chapter 5, along with the results obtained, regarding both the FM-WiFIX implementation, the hardware power consumptions and the RDS range. Finally, in Chapter 6, we review all the work done during this thesis, identify the main contributions and the future work.

Chapter 2

State of the Art

In this chapter, we present an overview on the different kind of approaches proposed so far to tackle the problem of energy inefficiency in WMSNs. We also attend to the impact of these approaches in the performance and throughput fairness, the other two major problems of WMSNs. Moreover, we present some specific solutions that are particularly suitable for the WVSNs paradigm, which is the focus of this work.

We start this chapter by presenting similar approaches to the one that will be followed in this thesis: the implementation of a low power out-of-band control channel to improve a system's energy efficiency. In Section 2.2 we present the current solution in the state of the art to solve the energy problem induced by Wi-Fi utilization, the IEEE 802.11 Power Saving Mode (PSM). Then, we go up in the OSI stack, starting by the existing solutions for energy-efficient medium access control at the data link layer in Section 2.3. Section 2.4 describes the energy-efficient routing techniques approach, a network layer solution. Finally, in Section 2.5 we show the importance of proper video encoding at the application layer, to reduce the overall power consumption of the network.

In Chapter 3, an extension of the state of the art is presented, focused only on the FM-WiFIX solution that we aim to implement with this work, and its related solutions and technologies.

2.1 Out of Band Signaling Solutions

The core of this thesis is the implementation of the low power out-of-band, FM-RDS based control channel, to allow shutting down the Wi-Fi radio interfaces of the sensor nodes when they are in idle state. Our channel must be energy efficient, reliable, capable of reaching every node in a large mesh network and must have a decent bandwidth.

This technique of using out-of-band synchronization signaling is not new and was already proposed multiple times in the state of the art, in several ways and for different contexts. Here, we present some solutions proposed specifically for increasing the energy efficiency of networks composed by wireless devices.

To start, we find important to refer one of the first studies about this thematic of power saving within an ad hoc network of sensor nodes. The design methodology for PicoRadio Networks, presented in [15], explains several methodologies that should be followed in the design of these networks, to achieve energy efficiency. The concepts referred are important and may be applied in our work. On the other hand, the specifications about the technologies that could be used to achieve this power efficiency are somewhat outdated.

Back in 2002, when PDAs were in vogue and provided a practical way of accessing the Internet, thanks to their built-in Wi-Fi card, the energy inefficiency of Wi-Fi was already identified and was a problem to the lifetime of these devices. To address this problem, in [16] is proposed a solution where the device would turn off when it is in idle state and would wake up upon the reception of specific control messages, received through a secondary, low power, always on, control channel.

Now regarding the energy efficiency in sensor networks, a very interesting concept is proposed in [17]. It consists on a radio-triggered circuit, powered by the radio signals themselves. When there is no reception of suitable radio signals, the circuit is shut down and so is the sensor. When the circuit is triggered, it also activates a wake-up interrupt, which awakes the sensor node. This is a suitable solution for sensor networks which only sporadically transmit data, not for a surveillance network of distributed video sensors, constantly transmitting video streams.

In [18], a similar concept of waking wireless interfaces through an out-of-band control is presented. However, in this solution they use a reverse beaconing technique to turn off Access Points, when they do not have any client connected. The requirements for their control channel are not particular demanding, they need of a low power radio module with a coverage at least equal to Wi-Fi and, as they only send sporadic short messages through the control channel, their bandwidth requirements are not demanding as well. Therefore, this is a very different scenario of our own.

A prototype of a low-cost wakeup radio for the 868 MHz band, to enable multiple protocols of out-of-band signaling, is designed, implemented and tested in [19]. Such device could be useful for us to implement our out-of-band control channel, instead of using the FM-RDS technology. However, the coverage area provided is too low, they could only achieve a range of 2 m at the end of the implementation.

The Carnegie Mellon University (CMU) has also developed FireFly, a time synchronized real-time sensor networking platform [20], where they deploy a network of globally synchronized sensors, using an AM signal. The sensors access the medium using a time division multiple access (TDMA) scheme. Each node is scheduled to transmit and receive data on dedicated 5 ms time slots and are shutdown during the rest of the time. This system employs some other, more complex features but the essential is that they use an AM signal to keep all nodes synchronized, which is crucial in such a TDMA scheme. The FM-WiFIX solution employs a similar method, based on FM radio. However, FM-WiFIX is a much more centralized solution, which requires a more frequent control signaling, unlike this FireFly solution where the out-of-band signaling is just used to keep a distributed system synchronized.

Knowing the work already done about out-of-band signaling, we conclude that our goal is rather ambitious, as we intend to apply our low-power, out-of-band control channel into a system where we must be able to signal state transitions of the wireless cards every few milliseconds. However, this also supports the validity of this thesis and the validity of the FM-WiFIX proposal [2] since a control channel like the one that we propose to implement, where swiftness is crucial, was never achieved before.

2.2 IEEE 802.11 Power-Saving Mode

Whenever WMSNs have large bandwidth requirements, Wi-Fi is the obvious choice to assure communications on WMSNs. However, Wi-Fi radio interfaces are major energy consumers which can be a problem in scenarios where energy consumption is critical. To mitigate this inconvenient, the IEEE 802.11 standard specifies a Power Saving Mode (PSM) for the infrastructure/BSS mode (Basic Service Set with an access point) and the ad hoc/IBSS mode (Independent Basic Service Set without an access point) [21]. Its main objective is set the transceivers in sleep mode whenever they are not in use.

From all the solutions found in the state of the art to address the energy inefficiency problem, this is the most used and one of the most effective. Since the PSM is specified in the IEEE 802.11 standard, any Wi-Fi card has already the capability of activating the PSM itself. Moreover, this approach of setting the transceivers in sleep mode is quite effective as it tackles directly the cause of the energy inefficiency: the high consumptions of Wi-Fi radio interfaces. Thus, using PSM, significant energy savings can be achieved, as shown in Table 2.1.

Device	Mode	Power Consumption
Intel 5300	IDLE RX	820 mW
Intel 5300	SLEEP	100 mW
Intel 4965	SLEEP	46 mW

Table 2.1: Comparison of IDLE Rx and SLEEP power states for two Intel 802.11n NICs [14].

As infrastructure and ad-hoc modes have some differences from each other, the power-saving mode operation for each one is also slightly different. Herein, we will only focus in the operation for ad-hoc networks, since it is the paradigm that applies to our work.

2.2.1 IEEE 802.11 PSM for Ad Hoc Networks

In PSM, the time is divided in time periods called beacon intervals, as shown in Fig. 2.1. At the beginning of each interval, the synchronization by beacon process takes place, which is needed to make sure that all stations wake up at the same time. After this synchronization procedure, the stations announce to their neighbors the existence of pending data frames using the unicast Announcement Traffic Indication Messages (ATIMs). This announcements take place during the ATIM window. During this interval, all the stations stay awake, listening to the medium to check if there are packets to receive or informing other stations about packets for them. Data is exchanged in Data TX/RX window. Only stations that successfully exchanged a pair ATIM-ACK during ATIM window stay awake in this period, transmitting or receiving data. All the other stations enter in sleep mode for the rest of the beacon interval [21].

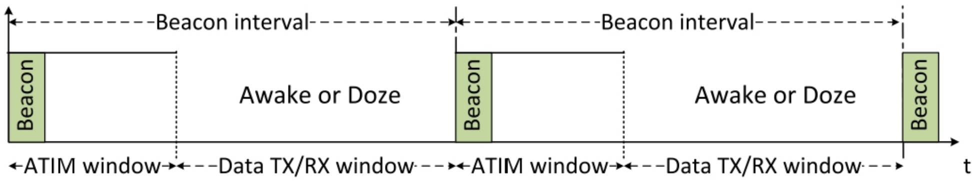


Figure 2.1: The IEEE 802.11 PSM time division into Beacon Intervals. [5]

This PSM protocol may be efficient for single-hop communications, for which it has been designed. However, for multi-hop networks, the described PSM scheme greatly increases the end-to-end delay in communications. Considering a typical multi-hop scenario where node A wants to send a packet to node D using nodes B and C as relays, in the best of chances, the packet will only arrive on D at the third beacon interval, since it takes one beacon interval for the packet to be transmitted from one node to another. Such a delay could be enough to substantially affect the QoS requirements of some applications. [5]

2.2.2 Multi-Hop Power-Saving Mode (MH-PSM)

To mitigate the problem previously described, an enhanced PSM for multi-hop communication was proposed in [5]. The strategy employed in the Multi-Hop PSM (MH-PSM) consists in sending, within the ATIM frame, not only the MAC address of the receiver and the sender but also the address of the packet's final destination. In the scenario where A needs to send a packet to D through B and C, when A sends the ATIM frame to B indicating that it has a packet to B, the ATIM frame would also contain the address of node D. Node B would realize that the packet that A is about to send is to be forwarded to D and so, it would check in its routing table the next-hop address to D and send an ATIM frame to C and C would do the same. Therefore, during the same data transmission window, the packet would be transmitted from A to B, then from B to C and finally from C to D, reducing the end-to-end delay from three beacon intervals to only one, as shown in Fig. 2.2

The experimental results of this solution proved that MH-PSM significantly outperforms the normal PSM, being their difference increasingly bigger with the number of hops. For six hops, MH-PSM presents an end-to-end delay near ten times smaller than PSM. An additional intra-beaconing mechanism called SoBT (Sleep on Beacon Transmission) was also proposed in [5]. SoBT presents a better scheduling of transitions between sleep and wake states, allowing stations to be in sleep state during bigger periods. With SoBT, the experimental results showed that the doze time ratio was approximately three times bigger when compared with the already increased

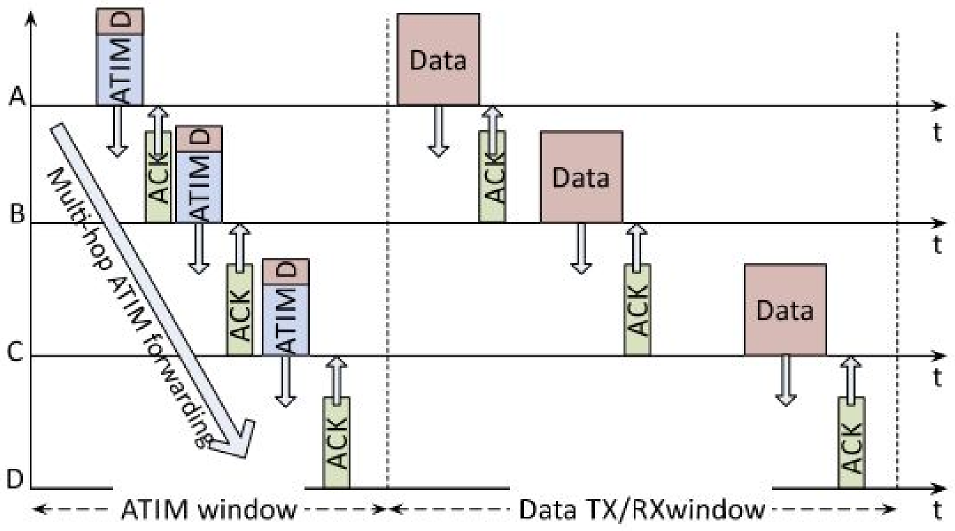


Figure 2.2: Using the MH-PSM, the transmission of a packet now only takes a single beacon interval instead of three with standard operation [5].

doze time ratio presented by MH-PSM. Thereby, this enhanced power saving mode for multi-hop networks not only reduces significantly the end-to-end delay but also increases the power savings.

This is undoubtedly an excellent enhancement to the standard PSM. However, both this solution and the original PSM have one major problem. In a scenario where nodes are continuously transmitting real-time data, such as in a WVSN, they would rarely enter in sleep state since they always have data to transmit. Thus, using PSM to improve our video monitoring system is not an option since it would not allow any significant energy savings. Instead, it would probably reduce even more the WVSN performance with the overhead introduced by the exchange of ATIM frames and beacon synchronization.

2.2.3 Other Optimizations to IEEE 802.11 PSM

In order to further improve the 802.11 PSM, several other optimizations have been proposed. Although the vast majority of them targets single-hop networks, there are also some solutions to optimize PSM in ad hoc networks, as reviewed in [5].

Some strategies employed aim to minimize the duration of idle listening by using mechanisms for early transition to the sleep state. As suggested by the values in Table 2.1, increasing the sleep time is an effective strategy to save power [14]. This could be done by modifying the ATIM announcements to include the number of pending frames to be transmitted. With this mechanism, as soon as the last expected frame is received, the node moves to the sleep state, without waiting for the end of the beacon interval. Other solutions try to improve this early transition even more, allowing the nodes to enter in sleep mode even before the end of the ATIM Window. However, all these solutions assume low-traffic scenarios and the additional energy savings are usually obtained at the cost of increased multi-hop end-to-end delay, which can be even worse than that of the standard 802.11 PSM.

Still, in order to increase the sleep time, there are other schemes proposed such as overhearing of the ATIM frames to decrease the ATIM Window, using the beaconing nodes as centralized schedulers, transit to the doze state or extend the active state beyond the beacon interval depending on the traffic. The latter also reduce end-to-end latency if the network load is high enough.

To address the problem of end-to-end latency there are also some solutions besides MH-PSM. These solutions rely on methods such as the knowledge of traffic patterns, cross layer coordination and prediction based on traffic history. MH-PSM outperforms any of these alternative optimizations referred.

After thoroughly investigating 802.11 PSM and all its proposed optimizations, it is clear that this is not a suitable solution for improving both the energy efficiency and the performance of WVSNs. Yet, it is a good protocol and the state of the art solution to save energy in Wi-fi based networks and some of these ideas could perfectly be used to further improve a solution designed specially for WVSNs.

2.3 MAC Protocols

To address the major problems and the special requirements of WMSNs, several solutions have been proposed to address this issues at the Medium Access Control (MAC) layer. It is important to assure that the MAC protocols used are energy-efficient, QoS-aware and try to maximize the network throughput.

The IEEE 802.11 standard itself has evolved in this sense. The legacy 802.11 MAC employed the Distributed Coordination Function (DCF) which relies on the CSMA/CA and on RTS/CTS mechanisms. Besides the bad performance of CSMA/CA in multi-hop networks explained in Section 1.2, this scheme did not cater any QoS support, being all packets treated evenly. Consequently, this mechanism was not suitable for real-time applications with high QoS requirements.

To overcome this problem, the IEEE 802.11e standard introduced a mechanism known as Enhanced Distributed Channel Access (EDCA) [22]. QoS support in 802.11e is provided by an innovative priority mechanism. In each station, packets are organized in several buffers according to their priority. When trying to gain the medium using the CSMA/CA scheme, the Arbitration Inter-Frame Space (AIFS) assigned for higher priority packets is smaller than the assigned for lower priority packets. Hence, a station with a high priority packet to send gains the medium and transmit the packet while the lower priority packets are still waiting in AIFS. Furthermore, when a node gains the medium in EDCA mechanism, it is granted with a Transmit Opportunity (TXOP) period during which it can send as many packets as possible, as long as the total access duration does not extend beyond a TXOP limit. Consequently, this new 802.11e features improve the performance and support QoS requirements. However, the priority scheme employed may worsen the throughput fairness since the access to the medium is rarely granted to lower priority packets.

To provide fairness and energy-efficiency, an optimization to the IEEE 802.11e was proposed in [22]. The MAC Protocol proposed, QEMAC, adds a dynamic duty cycling based on the RTS/CTS frames exchange and assures that lower priority packets are not forgotten in their buffers.

These solutions are suitable for WMSNs where different kind of traffic, with different priorities, coexist. However, for a WVSN where only video streams with the same QoS requirements are exchanged, these solutions are not very effective.

An interesting solution, HTSMAC, was proposed in [23]. HTSMAC is a cross layer solution based on a MAC protocol designed for Sensor Networks that would alternate between two modes: in normal mode the nodes are collecting measurements such as temperature, humidity or luminance. In image mode, nodes are transmitting images to a sink. When switching to image mode, a node would flood xSYNC packets. These xSYNC packets serve to inform the network to switch the MAC protocol for the one used in image mode and to inform the network layer to set up a chain topology from the source node to the sink. Every other node out of this chain would enter in sleep mode. When the transmission is finished, another xSYNC packet is sent to reestablish the normal topology and to wake up the neighbor nodes in sleep mode. This allows energy savings and avoids collisions. HTSMAC was proposed for Wireless Video Sensors that only occasionally collect images, thus not being suitable for WVSNs that are always transmitting video streams. Nevertheless, what is interesting here is the principle behind this solution, which is fundamentally similar to the one employed by FM-WiFIX, the solution that we aim to implement in this work.

A different approach to reduce the consumption of the IEEE 802.11 cards is followed in [24], where they minimize the energy consumption of idle listening by downclocking the Wi-Fi card during those periods. They assert that the Wi-Fi's energy inefficiency comes from the constant idle listening imposed by the CSMA/CA mechanism, to detect unpredictably arriving packets or to assess a clear channel. Thus, their idea is to put the radio in a *subconscious* mode during this idle listening period, without losing the capacity of promptly respond to incoming packets.

We can then conclude that the only strategy found among the state of the art MAC layer solutions, suitable for WVSNs, is the one already followed by FM-WiFIX. The downclocking approach could be useful from an implementation perspective. Shutting down the wireless interfaces is not an easy task and this could be an approximate substitute to the effective transition of the Wi-Fi card for the off state.

2.4 Energy-Efficient Routing Techniques

The utilization and optimization of energy-efficient routing techniques are the most popular approach found in the literature to address the energy inefficiency problem of WMSNs. They are relatively successful in extending the network lifetime in battery-constrained scenarios yet it is not possible to achieve an energy-efficient network by simply employing these routing techniques.

Per se, an efficient routing protocol that minimizes the number of collisions and hops to reach the sink would already provide some energy efficiency by reducing the overall number of transmissions. However, as presented below, there are a multiplicity of routing solutions that go further, aiming specifically to the energy issue.

These solutions follow different principles to tackle the energy inefficiency problem. The energy-aware routing protocols (Section 2.4.2) aim to maximize the network lifetime by making routing decisions based in the current energy level of each node or by choosing a path or multiple paths that will minimize the total amount of energy needed to reach the sink node. Hierarchical protocols (Section 2.4.1) divide the nodes into clusters to aggregate traffic and optimize the energy consumption in transmissions. Some try to adapt the QoS provided to the minimum required by discarding dispensable packets, thus saving energy with the avoided transmissions [4]. Others try to find a trade-off between higher transmission powers, which reduces the end-to-end delay at the cost of extra energy consumption, and the network lifetime. Many solutions try to combine some of these different principles. Independently of the strategy followed, the various solutions are always optimized for a given scenario or set of scenarios. Herein, we will mainly focus on the solutions suitable for a video streaming scenario, with its strict end-to-end delay requirements and extra bandwidth needs. To an overall vision on energy-efficient routing techniques for all WMSN scenarios, a fairly complete survey can be found in [4].

2.4.1 Hierarchical Routing Protocols

Hierarchical routing protocols divide the nodes into clusters. In every cluster, a node with higher processing power is selected as the cluster head and is responsible for aggregate traffic from the rest of the cluster nodes and send it, directly to the sink if possible, or to another cluster head closer to the gateway otherwise (Fig. 2.3). This result in a higher energy saving for the majority of network nodes, since they only have to transmit data to the respective cluster head. However, the cluster head will have higher energy consumptions, especially if transmitting directly to the sink over long distances and low quality channels, which is a usual situation.

To overcome this discrepancy, many hierarchical protocols use rotation of cluster heads to evenly distribute the energy load among the network nodes [25].

The Low-Energy Adaptive Clustering Hierarchy (LEACH) [26] is a good example of a self-organizing, adaptive clustering-based protocol that uses this rotation system. LEACH is a reference in clustering-based protocols, being largely cited in the literature. However, since LEACH was presented, in 2000, new protocols have been proposed to optimize LEACH. LEACH assumes that the cluster heads communicate directly with the sink. Nevertheless, as stated above, with this assumption, the cluster heads may spend too much energy communicating with the sink over large distances. Many proposed optimizations aim precisely at this question, establishing mechanisms to make cluster heads communicate with each other, as the situation represented in Fig. 2.3 and used in [25].

2.4.2 Energy-aware Routing Protocols

Energy-aware routing protocols use informations about the energy levels of the various network nodes when making routing decisions, in order to extend the network lifetime. When designed

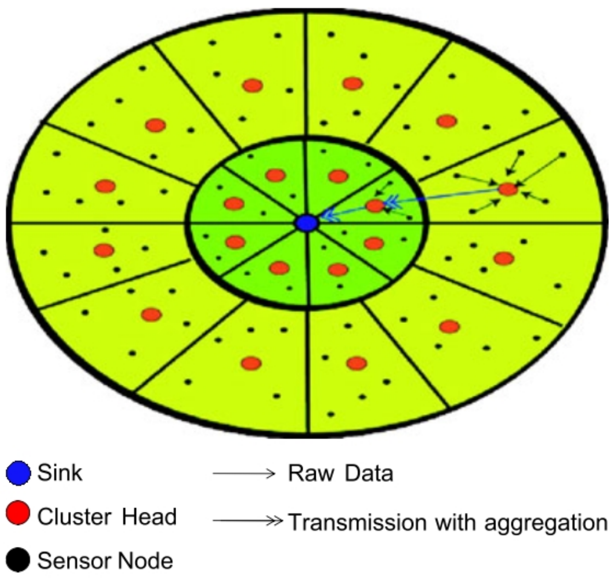


Figure 2.3: Clustered WVSN architecture [25].

to video stream scenarios, these protocols must also take in consideration the strict end-to-end latency and bandwidth requirements of WVSN.

The Optimized Energy-Delay Sub-Network Routing (OEDSR) [27] is one of these protocols, being a cluster-based, event-driven, multi-hop and energy efficient approach to the end-to-end delay constraint [4]. In OEDSR, the available energy, average end-to-end latency values of the links and the distance from sink are all considered when calculating the best next-hop forwarding node. This way, the protocol ensures that the chosen path from the cluster-head to the sink is loop-free, power-efficient and satisfies the QoS requirements regarding end-to-end delay. The protocol also considers the situation in which a node loses more energy than the rest. In this situation, an alternate path is calculated, thus maintaining the network load evenly distributed. OEDSR outperforms the well-known ad hoc On-Demand Distance Vector (AODV) in terms of end-to-end delay, collisions and energy consumption [4].

The Directional Geographical Routing (DGR) [28] is another interesting protocol designed to H.26L real-time video communications on WMSNs. In video transmission over sensor networks, the bandwidth requirements can be several times higher than the maximum transmission capacity of sensor nodes [4]. To address this problem, DGR divides the video stream into multiple sub-streams. Then, each of this sub-streams is sent towards the sink through a different path, using multi-path routing. This technique allows high delivery ratios and low end-to-end delays while making the best of the limited bandwidth and energy available. However, the performance of DGR can be compromised, since it assumes that any node can send video packets at any instant. Within large networks, this could rapidly cause the saturation of the system.

In [29] is proposed an energy efficient scheme that uses a modified LEACH protocol and a multipath video scheduling to minimize the video distortion in transmissions over WMSNs.

Similar to DGR, this scheme distributes the traffic load by selecting multiple paths, using the modified LEACH protocol. A further optimization that consists in adapting the QoS requirements to the channel capacity was also proposed. This is accomplished by dropping less important video packets. Therefore, the proposed power aware video packet scheduling is capable of minimizing the power dissipation across all network while keeping the perceived video quality at very high levels, even in extreme network conditions. [4]

Two algorithms for optimization of battery power in ad hoc networks are proposed in [30]. These algorithms work together in order to choose the node that requires less energy for a given transmission. The Select Node Algorithm is used to select all nodes with enough power to participate in data transmission. The Total Minimum Power algorithm finds the minimum energy required to send a given packet from node i to node j. This approach heavily relies on the distances between nodes and on the fact that a node with more power has a higher range than nodes with less power. This solution is presented here since it uses a strategy slightly different from the ones previously mentioned. However, this solution has not been optimized for WVSNs.

2.4.3 Mobile Sink

Although the vast majority of WMSN routing protocols do not take mobility into account, the sensors and sinks may be highly movable. In fact, there are evidences that adding mobile sensors in WMSNs could improve their performance, including coverage and energy efficiency [4]. Furthermore, the mobility of sensors and sinks could also improve real-time applications such as a video monitoring scenario.

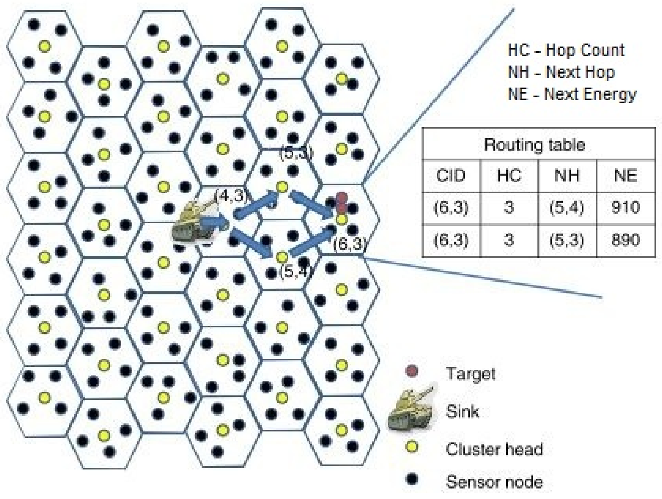
Usually this mobile sink approach is used jointly with the cluster-based networks referred in Section 2.4.1. In a hierarchical topology, the network lifetime is defined by the cluster heads physically closer to the sink since they are responsible for aggregating and relaying more traffic than the other nodes. The rotation of cluster heads are the commonly used measure to mitigate this problem.

The deployment of a mobile sink for target tracking applications could be another approach to maximize the network lifetime. The mobile sink could then be used to track a moving object. The efficiency of this strategy is highly dependable of the mobility pattern algorithm used. If effectively modeled, the mobile sink based solutions can significantly decrease the energy spent in relaying traffic [25] and provide better QoS, thus improving both the performance and the energy efficiency of the WVSN. The major drawbacks of this approach are the increased routing complexity and the extra energy spent on the transmissions of routing messages.

In [25] is proposed a node scheduling mechanism to improve the network lifetime and remove duplicated data. Still, in this paper, the performance of WVSNs was evaluated with a mobile sink versus a static sink and the performance of solar-powered video sensors to improve the system lifetime was also evaluated.

The routing scheme employed on this solution is illustrated in Fig. 2.4. All cluster heads build their routing tables from an initial ANNOUNCEMENT message broadcast by the sink. This message contains the hop count (HC), the Next Hop (NH) and the remaining energy of next hop (Next

Energy or NE). The ANNOUNCEMENT message is forwarded by cluster heads after updating the hop count. When receiving more than one ANNOUNCEMENT, the cluster head will select the path with the lowest hop count (HP). If multiple HCs are equal, then it will choose the route with the highest energy remaining (NE). Eventually, all cluster heads will have determined the shortest path to the sink. This process occurs each time the sink moves from one cluster to another. When the sink stays in the same cluster, this updating process is unnecessary. These routing table updates are needed to ensure that cluster heads never loose the connectivity with the mobile sink.



Routing table

CID	HC	NH	NE
(6,3)	3	(5,4)	910
(6,3)	3	(5,3)	890

Figure 2.4: Routing scheme with mobile sink. [25]

The sink objective is then intersecting the target. In a multiple target scenario, the sink objective is to move towards a position that represents the center of all targets.

The simulation results showed that the mobile sink model outperforms the static sink model for tracking applications. Jointly with the node scheduling mechanism, it also extends the network lifetime. The results also showed that the system performance could be greatly improved with the employment of rechargeable solar cells and unequal layered clustering algorithms.

2.5 Video Encoding

In order to reduce the energy consumption in transmissions/receptions over WVSNs, the video retrieved should be encoded or compressed before being transmitted to the gateway [31]. Even a simple coding algorithm, with low complexity, would make a significant difference in the bandwidth needed to transmit a given video stream when compared to transmitting the video uncompressed. By using efficient coding algorithms, with high compression rates, it is possible to sharply reduce the quantity of information to be transmitted, maintaining the perceptible video quality.

The widely used H.264 standard for example achieves typically a 50 percent reduction of average bit rate for a given video quality compared to MPEG-2 and about a 30 percent reduction compared to MPEG-4 Part 2 [32]. Its successor, H.265, is even more efficient, achieving about a 50 percent reduction on compressed video file sizes when compared to H.264, without any significant video quality degradation at the receiver [33]. These numbers show the huge impact of encoding in video transmission. Drastically reducing the amount of information transmitted would significantly reduce the overall energy spent in transmissions, specially in a multi-hop network where a video stream may be transmitted and received by several nodes before reaching the sink.

However, this improved energy-efficiency in transmissions comes at the cost of extra energy spent in each node to encode the video stream. The processing power required to run the encoding algorithms may consume a significant energy amount in a WVSN node. Thus, it is necessary to find a trade-off between the compression ratio of the algorithm used and its respective power consumption.

In [31] is presented a review of energy efficient block-matching motion estimation algorithms for WVSNs, being their objective to find the most energy efficient algorithm that retains image quality and can be applied to WVSNs. Since a motion estimation engine is the most costly computing block in video compression design [31], using the most efficient one would make a significant difference in the power consumption of the video compression algorithm.

[6] is an interesting article about the kind of system that we aim to improve, a wireless video sensor network for surveillance purposes, but focused on the power consumptions associated to specialized video sensors, specifically the video coding and analysis units. As in their experiments they isolate the power consumptions of the video sensors, this paper give us a very good idea of the power demands of an ASIC-based (application-specific integrated circuit) video sensor and how reduced that value is, in comparison with the Wi-Fi consumptions. Some of their results are depicted in Fig. 2.5, where P_a is the power consumption of the video analysis engine, P_c is the coding power, P_t is the transmission power and P_s is the sensor power. Note that, to isolate the video sensor's power consumption, they subtracted the background power measured when the development board is idle. This means that they also subtract the power consumptions of the Wi-Fi module attached to the board. Therefore, when they talk about transmission power, that value does not contain the power consumption of the Wi-Fi card, it only contains the processing power required by the sensor to transmit the video stream retrieved. That is why the power transmission value is a small portion of the total power consumption.

This coding approach does nothing to mitigate the cause of the energy inefficiency: the high consumptions of Wi-Fi interfaces. Nevertheless, it addresses the energy problem at the application layer, thus being a good addition to lower layers solutions such as the ones previously presented.

2.6 Summary

Throughout this chapter, the most relevant solutions to address the problem of energy inefficiency in WMSNs, and, specifically, in WVSNs were presented.

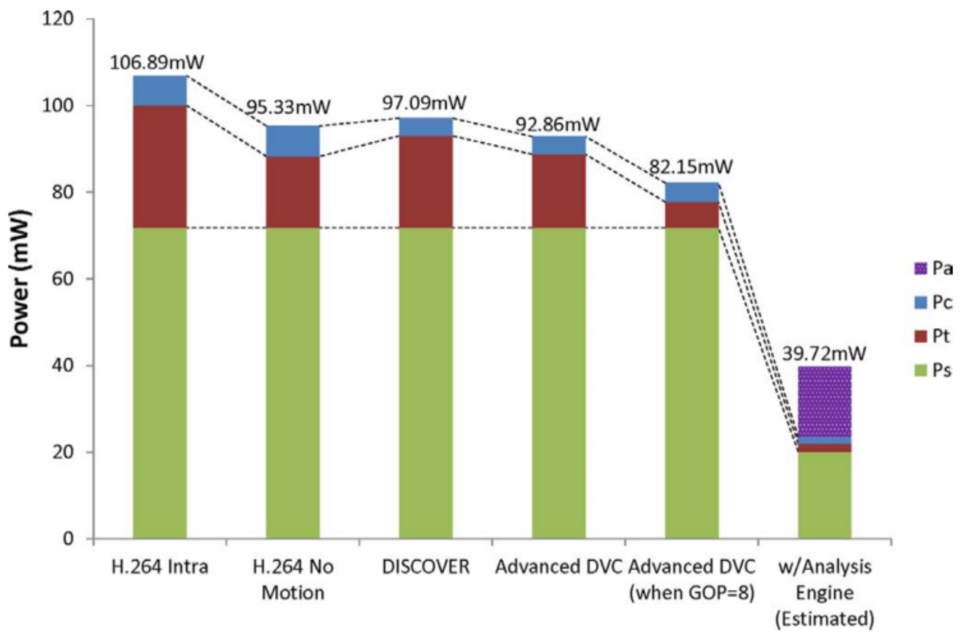


Figure 2.5: Power analysis of an ASIC-based video sensor node and the power analysis of several expected solutions [6].

We concluded that, despite the many existing solutions that use out-of-band control channels to keep a distributed system synchronized, a control channel with the characteristics of the one proposed in FM-WiFIX was never implemented before.

Regarding the state of the art IEEE 802.11 PSM, this is, with no doubt a valid and extensively used solution. Applying its enhanced version [5] to multi-hop networks would make this solution even more attractive. However, as explained in Section 2.2.2, within a WVSN scenario the nodes would rarely enter in sleep mode, canceling the improvements introduced by the PSM.

The Medium Access Control protocols for improving WMSNs operation present interesting energy saving mechanisms but none of them are suitable for a WVSN as the one we are interested in.

The routing solutions presented, although interesting for many scenarios, are not suitable for our work due to two main reasons. Firstly, they only address the energy problem at the network layer, using strategies such as selecting the next-hop node in function of the energy level of the neighbors. This kind of solutions extends the network lifetime but does not make the network itself energy efficient. The cause of energy inefficiency is mainly due to the high consumptions of Wi-Fi interfaces and without mitigating those consumptions, it is not possible to achieve an energy-efficient network. Secondly, these routing solutions overlook the performance and fairness problems of WMSNs. In this work, we are focused in achieving the best performance at the lowest energy consumption. Our goal is to implement an energy-efficient system and not extending a bit more the lifetime of an energy inefficient network. Ultimately, this kind of approach could be inserted in our implementation in a perspective of combining a multiplicity of solutions. However, the probable energy gains do not seem appealing enough to justify the additional complexity. The mobility sink model is the only solution that might be worth to implement, in a WVSN for target tracking.

The encoding approach consists in an application layer solution, thus being independent of the previous discussed solutions. It is useful to increase even more the energy efficiency of the network but in this work our focus goes to the lower layers of OSI model (physical, data link and network layers). Therefore, a deeper analysis on this kind of solutions is out of the scope of this thesis.

Taking into account the solutions found on the state of the art to address the three major problems of WMSNs/WVSNs, we can conclude that first, an extensive work has been done regarding WMSNs but few of those solutions are appropriate for WVSNs. Secondly, from the few solutions suitable for WVSNs, none of them is a complete solution, they only address one or two of the three major problems at a time.

To the best of our knowledge, there is not any other complete solution as the presented by FM-WiFIX. In the next chapter we provide a more detailed explanation about this proposal and related solutions.

Chapter 3

The FM-WiFIX Solution

As the main purpose of this thesis is to implement the FM-WiFIX solution, proposed in [2], this chapter details the main theoretical aspects regarding this proposal. However, FM-WiFIX is not a stand-alone solution. It was proposed to complement two solutions already implemented, WiFIX and PACE. While these two existing solutions address the poor performance and the throughput unfairness in WMSNs, the new FM-WiFIX proposal tackles the energy inefficiency problem. Being implemented as a single technology, this set of solutions stands out from the ones presented in the State of the Art since it addresses all the three main problems of WMSNs at once. Therefore, we also describe these two companion solutions and provide some details about their current implementation, which will be adapted to also implement FM-WiFIX.

3.1 FM-WiFIX Overview

To address the problem of energy inefficiency, an innovative solution, FM-WiFIX, was proposed in [2]. FM-WiFIX is a scheduling mechanism that uses FM-RDS as a control channel to instruct the network's nodes to switch on /off their Wi-Fi interfaces, according to necessity. Similar to IEEE 802.11 PSM, the objective is to move the nodes to a power saving state whenever they are not needed to transmit, receive or relay data. However, with FM-WiFIX, instead of only moving the Wi-Fi interface to sleep mode, it could be possible to actually shut it down, thus completely eliminating the Wi-Fi energy consumption when the node is in idle state. Since a FM transmitter/receiver consumes approximately ten times less energy than a Wi-Fi card in sleep mode [2], FM-WiFIX is by far a more interesting solution to increase energy savings. Moreover, as FM-WiFIX uses an out-of-band control channel, it also eliminates overheads such as ATIM frames in 802.11 PSM.

To implement FM-WiFIX, it is assumed that WiFIX is used, in first place, to automatically create a logical tree topology rooted at the sink node, as illustrated in Fig. 3.1. As demonstrated in [12], this is a well-suited routing solution for a video monitoring scenario where the majority of traffic is exchanged from the video sensors to the sink node. FM-WiFIX is also built upon PACE, a scheduling mechanism proposed in [11], as an evolution to WiFIX, to address the performance issues of WMSNs caused by hidden nodes and throughput unfairness. Being two of the three major problems solved by PACE, FM-WiFIX addresses the energy inefficiency problem using a FM-RDS based control channel to turn on and off the Wi-Fi radio interfaces. Thereby, the FM-WiFIX's objective is to mitigate the energy inefficiency problem while keeping the performance enabled by PACE and the backwards compatibility with legacy Wi-Fi devices [2]. Since FM radio signals have a great coverage area, only one FM transmitter, located at the gateway, is expected to be enough to cover the entire WMSN area, as shown in Fig. 3.1. Thus, this solution only requires the deployment of an extra FM receiver for every WMSN node and the above-mentioned FM transmitter at the gateway.

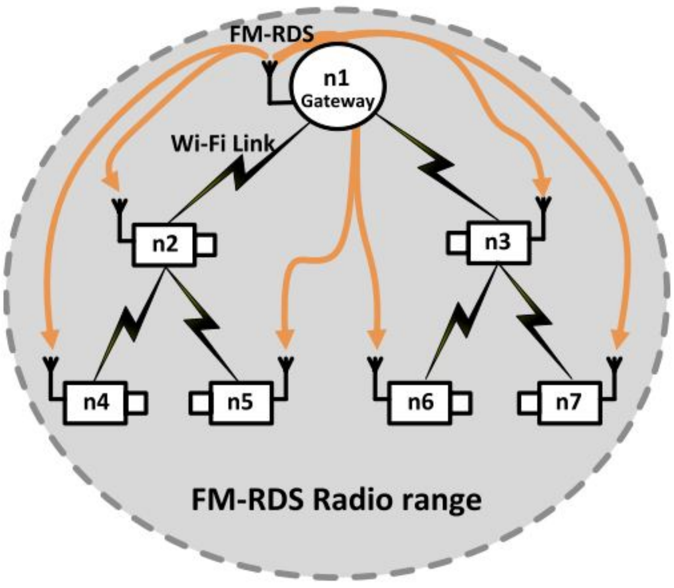


Figure 3.1: FM-WiFIX reference scenario [2].

In order to provide extra enlightenment about this solution, we present below a more detailed description about the technologies employed by FM-WIFIX and their operation.

3.2 WiFIX

The Wi-Fi Network Infrastructure eXtension (WiFIX) is a simple and efficient routing solution proposed in [12] to extend Wi-Fi wired infrastructures.

The solution proposed is rather simple. To handle the frame forwarding inside the Wireless Mesh Network (WMN), WiFIX reuses well-known concepts such as IEEE 802.1D bridges and their learning mechanisms. The network self-organization is ensured by a single-message protocol called Active Topology Creation and Maintenance (ATCM). This ATCM creates an active tree topology, rooted at the Master Mesh Access Point (MAP), the AP connected to the wired infrastructure.

The reason that motivated the WiFIX development was bypassing the limitations and complexity of the existing solutions to enable pervasive Internet Access in a WMN scenario, like the one illustrated in Fig. 3.2. As demonstrated in [12], WiFIX performs well in the scenario presented, where the majority of traffic is exchanged between MAPs and external networks through a unique gateway, the Master MAP.

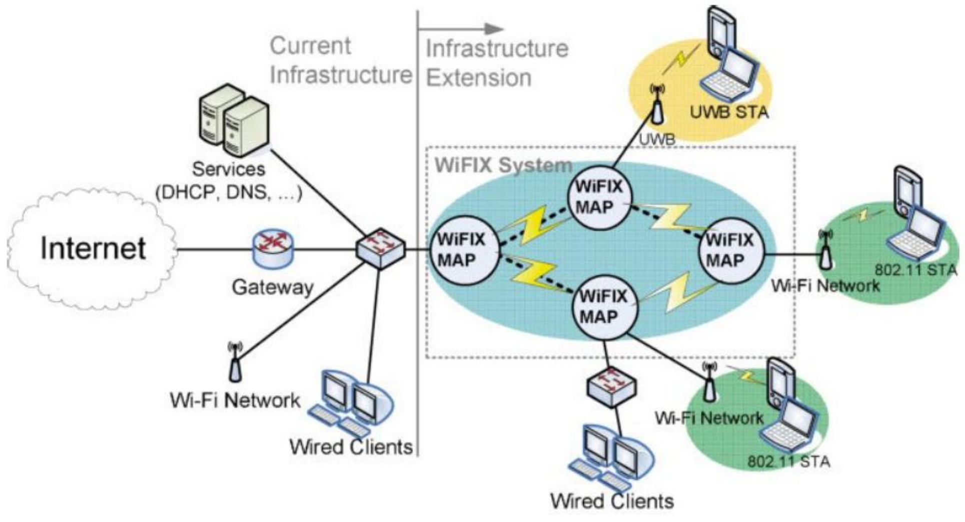


Figure 3.2: WiFIX reference scenario [12].

In a video monitoring scenario, instead of a WMN formed by multiple MAPs, we have a WMSN formed by multiple video sensors. The traffic between nodes is also assumed to be null. As these two situations are virtually the same, we can assume that WiFIX is also suitable for the scenarios where FM-WiFIX will be implemented.

3.2.1 WiFIX Implementation

Due to its needs of creating and deleting virtual interfaces and an 802.1D software bridge, WiFIX was implemented using the Linux Operating System, which provides the required tools. The virtual interfaces behave as Ethernet devices for the Linux bridge and are added to it, acting as one big network [12, 34]. The interactions between the modules and interfaces involved in the WiFIX implementation are represented in Fig. 3.3.

The *tap* interfaces, provided by the *vtun* module, are virtual interfaces that behave as an end-point to the tunnels established between each MAP and its neighbors. As aforementioned, these virtual interfaces are seen as Layer-2 Ethernet devices by the upper layers.

The interactions between these modules are as follows. At the master MAP, when a frame (e.g. from the Internet), arrives through an interface, connected to the learning bridge, the bridge will know to which *tap* the frame must be sent. From the *tap*, the frame will be delivered to the correspondent file descriptor, on the WiFIX user space application, that created the *tap*. The frame will contain an Ethernet header, with the MAC addresses of its source and its final destination.

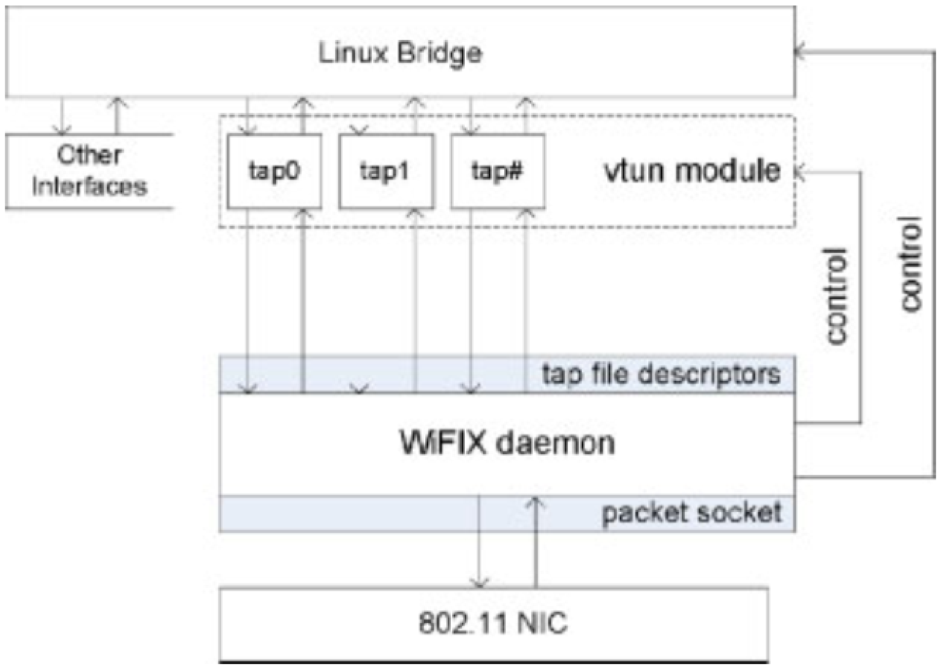


Figure 3.3: Interaction between WiFIX and its peer Linux modules [12].

It is up to the WiFIX daemon to process the frame and choose its next hop, to reach the final destination. When the master MAP receives a data frame through the wireless NIC, the WiFIX daemon will deliver it to the learning bridge, which will then deliver the frame to the correct interface.

From the perspective of a normal MAP, if a data frame is received on its downward path, through the wireless NIC, WiFIX will look to the final destination address, in the frame's inner Ethernet header, and update the outer Ethernet header with the next hop address for that destination. Then the frame is immediately delivered to the wireless NIC to be forwarded. When a data frame is received on its upward path, then the MAP immediately forwards it to its parent. If the final destination is the MAP itself, then the frame is delivered to the learning bridge, which will then deliver it to the correspondent terminal. When the MAP receives a frame from one of its terminals, a process similar to the one described when the master MAP receives a frame from the Internet occurs.

To update the network topology, the WiFIX daemon periodically runs the ATCM mechanism in order to detect the parent and child nodes and creating the active tree topology, thus assuring a loop free topology. It is during this process that *tap* interfaces are created. Each node creates a *tap* interface per neighbor and add them to the bridge. Each tap is associated with a file descriptor and that association is kept in a table, within the WiFIX daemon.

3.3 PACE

As stated before, there are three major problems associated to WMSNs: bad performance, throughput unfairness and energy inefficiency. To address the first two, the PACE solution was proposed in [11].

With CSMA/CA alone, the nodes access the medium in an uncoordinated way, causing the number of collisions to greatly increase with the network load and risking the network operation. Thus, the PACE mechanism aims to provide coordinated access among nodes to prevent collisions, without requiring any complex synchronization [11].

PACE was proposed as an evolution to the network extension solution, WiFIX. Therefore, it assumes that a logical tree topology rooted at the WMSN gateway is created over the psychical network using WiFIX [12, 11]. Within this scenario, the gateway acts as a central controller, providing the desired coordination to the network nodes. This control is two-fold. Firstly, it limits the transmissions to a single node at each time, avoiding collisions. Secondly, it ensures that every node has the opportunity to send a frame at each transmission round, thus enabling fair access to the medium [11]. Thereby, PACE solves simultaneously the problem of bad performance and throughput unfairness in saturated networks.

Moreover, within the FM-WiFIX solution, the performance boost introduced by PACE is further improved. It is well known that control frames, although small, degrade the system's performance [11]. Nevertheless, within FM-WiFIX solution, PACE's polling mechanism is implemented using an out of band control channel - the Radio Data System (RDS), in the FM band - thus improving the system's performance by eliminating the overhead of the control messages.

A possible drawback of this scheme is its utilization in scenarios where the traffic generated is heterogeneous. This is, in a network where each node has different bandwidths requirements, the fair medium access provided by PACE will deteriorate the network performance, since equal opportunities of transmission are being given to all nodes when they have different amounts of data to send.

3.3.1 PACE Implementation

The PACE implementation was developed along with the existing WiFIX program. PACE added to WiFIX the scheduling mechanism with the polling messages; the high priority messages, used for ARP and DHCP messages; the MAP registration process; circular lists, where frames to forward to MAPs or terminals are kept if they cannot be immediately sent; and the management of the terminals, relatively to the MAPs.

The current PACE implementation will be the starting point to the development of our mesh network of surveillance cameras, based on the FM-WiFIX proposed solution.

3.4 FM-RDS

The Radio Data System (RDS) is a technology used to transfer digital data over sidebands along-side FM radio broadcasts (Fig. 3.4).

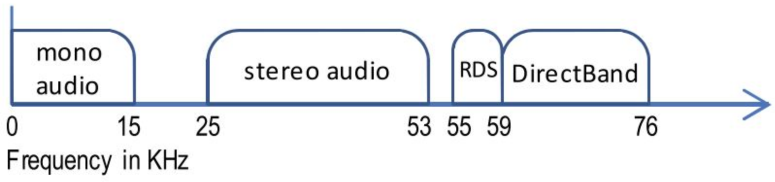


Figure 3.4: Right half side of demodulated FM radio spectrum [35].

Since FM radios are ubiquitous and a vast majority of them are prepared to decode RDS, it's easy and inexpensive to receive and interpret RDS signals [35]. Moreover, FM transmitters and receivers have low energy consumption, thus being an interesting solution for applications where energy consumption is critical.

On the other hand, RDS provides very low bit rates. As stated in the standard [36, p. 8], the RDS bit rate must be precisely $1187.5\,\text{bit/s} \pm 0.125\,\text{bit/s}$. Excluding the bits that are used in each block for the checkword and for the offset word (10 per block, 40 per message), this results in a maximum goodput of $730.8\,\text{bit/s}$:

$$104\,\text{bit} - 40\,\text{bit} = 64\,\text{bit} \Rightarrow 1187.5\,\text{bit/s} \cdot \frac{64\,\text{bit}}{104\,\text{bit}} = 730.8\,\text{bit/s} \tag{3.1}$$

Therefore, the amount of data that can be transmitted using RDS is very limited. The RDS standard deals with this limitation by using multiple messages to transmit a given field. For example, to transmit the entire Program Service (PS) name, a total of four type 0A groups are required [36, p. 18].

Field-testings showed that the mobile reception of RDS messages was more reliable when the radio data streams were broken into small frames [37]. Therefore, each RDS message has a size of 104 bits and is composed by four blocks of 26 bits each, as shown in Fig. 3.5.

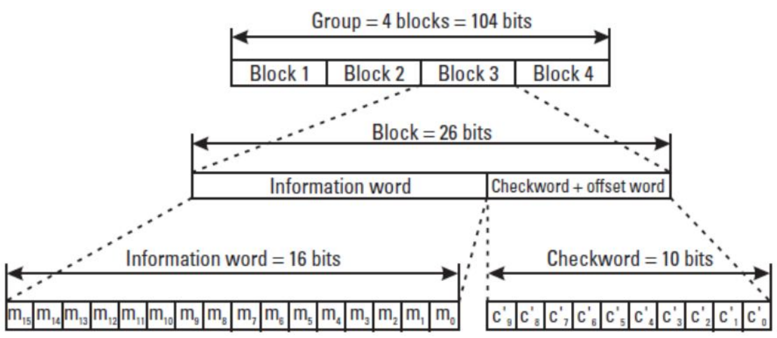

Group = 4 blocks = 104 bits
Block 1
Block 2
Block 3
Block 4
Block = 26 bits
Information word
Checkword + offset word
Information word = 16 bits
Checkword = 10 bits
m_{15} m_{14} m_{13} m_{12} m_{11} m_{10} m_9 m_8 m_7 m_6 m_5 m_4 m_3 m_2 m_1 m_0
c'_9 c'_8 c'_7 c'_6 c'_5 c'_4 c'_3 c'_2 c'_1 c'_0

Figure 3.5: Structure of the RDS baseband coding [36].

Each block contains a 2 B information word, where the data actually goes, an offset word used for group and block synchronization and a checkword, used for error-correction and detection. The checkword and the offset word are used together to generate the 10 check-bits of each block [36].

In the RDS standard, the bit stream is transmitted continuously so, although the existence of well-defined message types, there is no such thing as a "message" sent at one point in time (thus the need of offset words for synchronization purposes).

In FM-WiFIX, the interest in RDS does not go beyond physical layer. The reach of radio broadcasts in the FM band and the low power consumption of FM transmitters/receivers are the RDS characteristics that matter for the FM-WiFIX solution. The data-link layer aspects of RDS such as the frame structure and the various message types are irrelevant for this usage.

3.5 FM-WiFIX Scheduling Mechanism

The scheduling mechanism employed by FM-WiFIX works as follows. First, after the automatic creation of the tree rooted at the sink node by WiFIX, each node registers at the gateway. Herewith, the gateway learns the network topology. After the registration process, the gateway will have L vectors V_j, one for every leaf node j. These V_j vectors contain the leaf node j and all the relay nodes between j and the WMSN gateway. With this information, the gateway will then, for every V_j, send a message with the MAC addresses of all nodes i that must turn ON their Wi-Fi interface. Then, for every node i, starting from the leaf node, the gateway will set a timeout $T_{i,j}$ and wait until a message from i is received or the timeout $T_{i,j}$ expires. After this waiting time, the gateway will send a message to node i turn OFF its Wi-Fi interface and start a new iteration with the next node i in V_j [2].

Algorithm 3.1 FM-WiFIX scheduling algorithm running in the gateway [2].

```
1: **for all** $V_j, j \in [1..L]$ **do**
2:     send FM-RDS message to switch ON the Wi-Fi radio of all nodes $i \in V_j$
3:     **for all** $i \in V_j$ **do**
4:         set $T_{i,j}$
5:         **while** frame from *node i* is not received **OR** $T_{i,j} > 0$ **do**
6:             Decrement $T_{i,j}$
7:         **end while**
8:         send FM-RDS message to switch OFF Wi-Fi Radio of *node i*
9:     **end for**
10: **end for**
```

Chapter 4

Development

This chapter presents the implementation process of the FM-WiFIX solution. We explain how the original, theoretical proposal was changed throughout this work and what motivated those changes. We also describe the process of integrating this new solution with the existing implementations of WiFIX and PACE, the control protocol created and its correspondent implementation, using RDS as the control channel.

4.1 Changes to the Original Scheduling Mechanism

The original FM-WIFIX scheduling mechanism, proposed in [2] and summarized in Section 3.5, consisted in sending a message containing the MAC addresses of all nodes, in a given branch, that should turn the Wi-Fi radio on. After the successful reception of a packet from the leaf node of that branch, or a timeout, the gateway would send another RDS message, telling the leaf node to switch its Wi-Fi interface off and the transmission opportunity would be given to the parent of the leaf node. This would continue until the end of that branch was reached and then, the algorithm would move on to another branch (see Algorithm 3.1).

4.1.1 Major Problems of the Original Proposal

There are three main problems with this approach. Firstly, this method implies the transmission of several MAC addresses in each control message. A MAC address has 6 B (48 bit). As demonstrated in Eq. (3.1), the RDS maximum goodput is 730.8 bit/s. This could be a problem as it would slow down the RDS polling and cause it to be more susceptible to errors. Secondly, the partial loss of the fairness provided by PACE. In the original scheduling mechanism, the polling is done by branch and not by node. This means that nodes which belong to more branches will have more transmission opportunities. Lastly, with this mechanism, there isn't a specific signaling message to poll a node, after its child's transmission. The algorithm only states that after receiving a message from *node i*, the gateway sets the timer $T_{i+1,j}$ and waits for a message from *node i+1* (see Algorithm 3.1). There is no reference to how the *node i+1* knows when it should start its own transmission.

4.1.2 Modifications Proposed

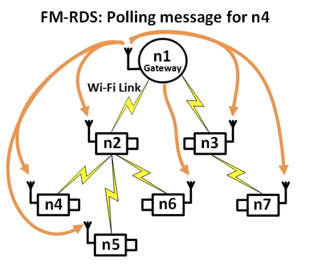
In order to improve the original solution and achieve a better WVSN, we propose three major modifications to the suggested FM-RDS based control mechanism. To minimize the amount of information to be transmitted, thus minimizing the first problem described above, the gateway assigns to each node, at the register time, an 8 bit identifier (allowing a network of 256 nodes). The assignment of these IDs avoids the use of MAC addresses, six times bigger, which considerable improves the efficiency of the control protocol. The intermediate nodes between a given node and the gateway learn the ID's of their descendants by snooping the message sent by the gateway, where this 8 bit identifier is assigned to the newly registered node.

Secondly, to clearly signal when a node should start to transmit, we propose to mimic the PACE's polling mechanism, i.e., sending a RDS message every time we wish to poll a new node, containing only the identifier of the node that gained the transmission opportunity. The intermediate nodes, when receiving a polling message addressed to one of its children, would also turn on their Wi-Fi interfaces, ensuring a path of transmission from the transmitting node to the gateway. Similarly, when hearing a polling message not addressed either to itself or to one of its children, the node would switch off its Wi-Fi interface, thus suppressing the need of control messages to turn the Wi-Fi off.

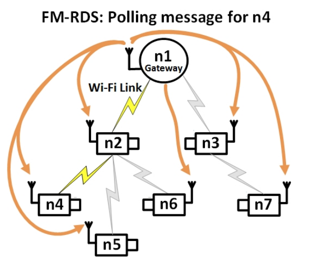
Finally, to address the loss in throughput fairness, we need to change the polling order such that each node is polled exactly once every round and the Wi-Fi state changes are minimal (the toggling between on and off introduces some overhead, which we want to reduce to a minimum). Since our network is organized as a tree, a suitable manner of achieving a polling order with the listed properties is using the post-order tree traversal. Fig. 4.1 exemplifies this new scheduling mechanism by showing how the nodes switch their Wi-Fi interfaces on and off, according to the control message being broadcast and also by showing the new polling order. The depicted behavior continues to repeat itself during the system operation.

With these modifications, the fairness provided by PACE is restored and the reception of the RDS message, by the polled node, act as a trigger for the beginning of the data transmission. Moreover, the network becomes more scalable and the control channel is able to switch on/off the nodes at faster rates, which would help to prevent the scheduling mechanism from being a bottleneck in data transmissions.

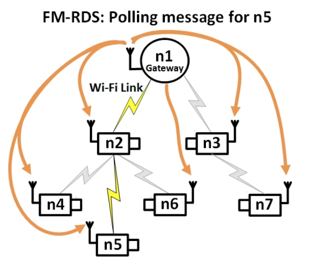
Along with the stated advantages, these modifications introduce new requirements into the system. The utilization of 8 bit identifiers introduces a limit of 256 nodes in the network. However, this limit could easily be exponentially increased, by using an additional byte, without significant impact in the control protocol, even though a WVSN with 256 nodes is already a very unlikely scenario. This new mechanism also implies that each sensor node knows its ID and their children's ID. The gateway must know the entire network topology.



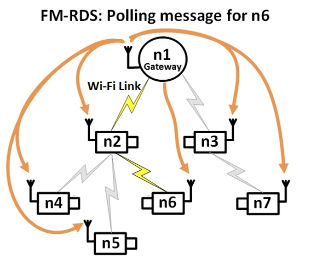
(a) Beginning of the FM-RDS control. All nodes have their Wi-Fi on and the first node polled is n4.



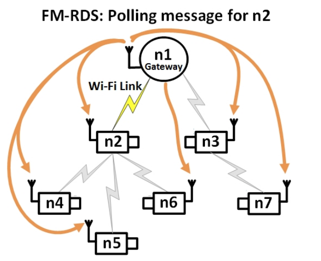
(b) Upon reception of the RDS polling message to node 4, only n4 and n2 remain with their Wi-Fi on. The others turn it off as they are not needed for transmit, receive or relay data.



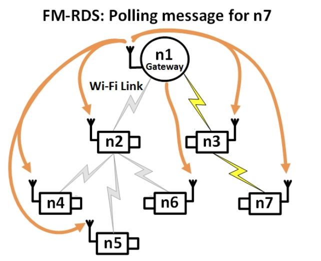
(c) As n5 receives a polling message addressed to its ID, it turns its Wi-Fi on. Since n5 is not a descendant of n4, n4 turns its Wi-Fi off now.



(d) n6, the last descendant of n2 is now being polled. Similarly to the last two scenarios, only n2 and its child being polled are with the Wi-Fi on.



(e) n2 is at 1 hop from the gateway so, when the polling is addressed for it, only n2 has the Wi-Fi on.



(f) The next node to poll, following the post-order tree traversal, is n7. Therefore, n7 turns its Wi-Fi on and n3, as its parent, does the same.

Figure 4.1: Video sensors switching their Wi-Fi interfaces on and off according to the RDS control messages broadcast. After n7, the gateway would poll n3 and then, would return to 4.1b.

4.2 Implementation of the Modified FM-WiFIX Solution

As previously explained in Chapter 3, the FM-WiFIX proposal is supposed to complement the already implemented WiFIX and PACE solutions, making up a single solution to address the three main problems of Wireless Multimedia Sensor Networks (WMSNs) at once. The current implementation of PACE, which already includes WiFIX, was our starting point to accomplish this goal. By modifying some parts of it and integrating a whole new set of functions, that implement the FM-WiFIX logic into the existing program, we turned the PACE implementation into a functional new program that integrates the three solutions, the FM-WiFIX solution. The rest of this section describes the changes and additions that were made to PACE, in order to implement this new solution.

For a better understanding of the following explanations, it is important to remember that, in WiFIX and PACE, we have a Wireless Mesh Network (WMN) of access points, called Mesh Access Points (MAPs). In our scenario, we have a Wireless Video Sensor Network (WVSN) where the former MAPs are video sensors (and do not have terminals attached) and the Master MAP is the gateway node.

4.2.1 Register Phase

In order to keep the network topology up to date, in WiFIX, the master MAP (in our context, the gateway node) periodically sends Topology Refresh (TR) messages (called hello messages on the implementation). The other MAPs forward this messages downward, after updating some fields such as the parent address or the distance to the gateway [12]. Based on the multiple hello messages received, when the topology refresh timeout expires, each MAP chooses a neighbor. If it chooses a new neighbor, a registration message is sent to the gateway. On the other side, if a dead MAP is detected, a deregistration message is also sent to the gateway. If the MAP maintains its current neighbor, no action takes place. This registration process is presented in Fig. 4.2. The field that contains the gateway's MAC address has no use right now but it would be required in a scenario with multiple gateways (addressed as future work on this set of solutions).

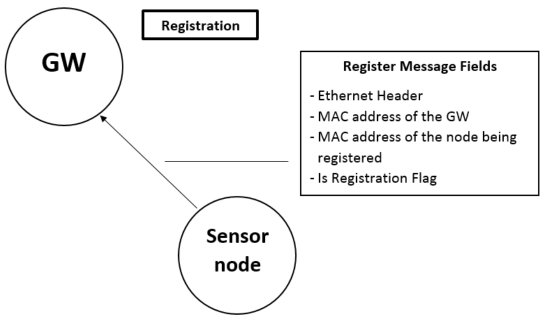


Figure 4.2: The content of the registration messages in PACE's original implementation.

In order to meet the requirements introduced by the new scheduling mechanism, described in the previous section, we had to perform several changes to this registration process.

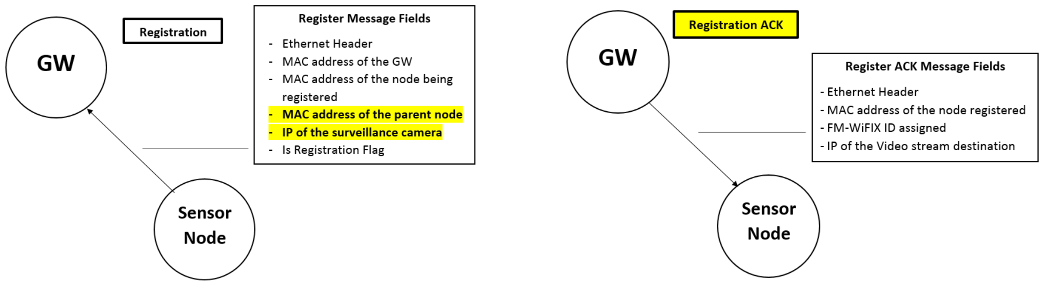
To enable the learning of the network topology by the gateway, now a node must include on its register message the MAC address of its parent. With this additional information, the gateway is now able to know the entire network topology.

To assign the 8 bit identifiers, we had to introduce a new message, the registration acknowledgment message. This message is sent by the gateway after registering a new node (or updating the information about an existing one) and must contain both the MAC address of the node registered and the FM-WiFIX ID assigned. On its downward path, the relay MAPs snoop this message and save the FM-WiFIX ID on their descendants list. This way, when hearing a RDS polling for one of their child, they know that they must turn their Wi-Fi on, as the ID being broadcast is on their descendant list. The purpose of this acknowledgment message is threefold: it assigns an ID to the registered node; it allows intermediate nodes to learn their children IDs; and it confirms a successful registration, thus increasing the reliability of the process. If the registration was unsuccessful, the node will not have an ID assigned. Therefore, when choosing a neighbor, if the previous one is maintained but the node has not an ID yet, it will send a register message again, instead of doing nothing as in the original PACE. This guarantees that the node is registered.

This register phase suffered one last modification due to a problem related with the Address Resolution Protocol (ARP). On PACE implementation, ARP requests and ARP responses are high priority messages, that are immediately sent, independently of the node which has the transmission opportunity. This is a good solution since it allows an immediate resolution of the IP address and the CSMA/CA is perfectly capable of dealing with the two simultaneous transmissions (the node transmitting its data and the ARP request/response). However, in our context, it is highly probable that the node that should answer to the ARP request is with its Wi-Fi off at the time. As a result, the IP address would remain unresolved. In order to avoid this kind of problems and having in mind that, in our context, we only have sensor nodes (with the role of MAPs) and no terminals attached, we chose to use static ARP entries. Each MAP has a wireless interface with no IP that is used in the layer 2 communications of the WiFIX-PACE logic and a camera associated to the IP address of a different interface (specifically, we associate the camera to the IP address of the bridge interface, created by PACE-WiFIX). The gateway uses the same scheme to associate the application that aggregates the video streams to an IP address (in the same network of the cameras).

In this scenario, our program can easily know the IP address associated with the video stream generated by each camera (at sensor nodes) and the IP address of the application that aggregates this video streams (at the gateway). Therefore, the sensor nodes send the IP address of their surveillance cameras in the register message and the gateway sends the IP address of the video application in the register acknowledgment message. Then, upon reception of these messages, the gateway creates a static ARP entry for each node, mapping the camera's IP address with the MAC address of the node and each node creates a static ARP entry mapping the IP address of the application with the gateway's MAC address.

This new registration scheme is depicted in Fig. 4.3.


GW
Registration
Register Message Fields
- Ethernet Header
- MAC address of the GW
- MAC address of the node being registered
- MAC address of the parent node
- IP of the surveillance camera
- Is Registration Flag
Sensor Node
GW
Registration ACK
Register ACK Message Fields
- Ethernet Header
- MAC address of the node registered
- FM-WiFIX ID assigned
- IP of the Video stream destination
Sensor Node

(a) The content of the modified register message. (b) The content of the new register acknowledgment message.

Figure 4.3: The differences between the new registration protocol and the previous one are highlighted.

4.2.2 Polling Order

In PACE, once a node is registered, it is added to an array that contains all the nodes registered at the gateway. When a transmission opportunity ends (either by the successful reception of a packet from the requested node or by the expiration of a polling timeout), the algorithm just picks the next node in that array. As the nodes in the array are placed as their registration is received, the polling in PACE does not follow any specific order, it is almost random. As previously mentioned in Section 4.1, in FM-WiFIX we must poll the nodes so that the Wi-Fi state changes are minimal. A suitable solution for this problem is to poll the nodes following the post-order tree traversal (see Fig. 4.4).

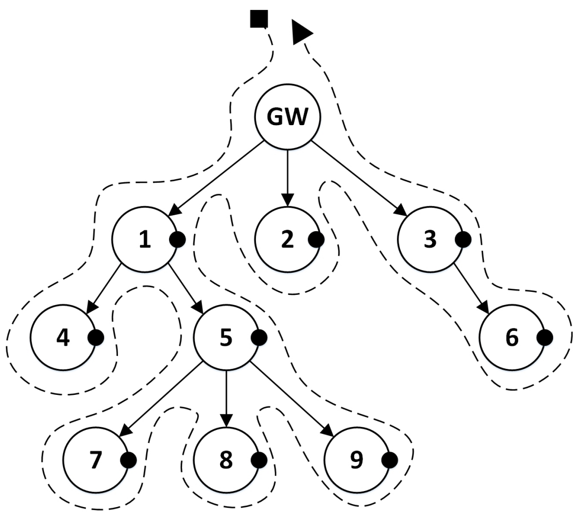


Figure 4.4: An example of a post-order tree traversal for an unbalanced tree. The polling order in this network would be: 4, 7, 8, 9, 5, 1, 2, 6, 3.

The implementation of this algorithm for binary trees is straightforward and is widely documented on the Internet. For non-binary trees, its implementation is more complex. Therefore, we developed Algorithm 4.1 to implement the post-order tree traversal for FM-WiFIX trees, using the knowledge about the network topology available at the gateway, which is the parent of every node registered. The function depicted in the algorithm is recursive and calculates the polling order for a given tree (rooted at *node_id*). Thus, the first time it is called, we must pass the gateway's ID as parameter. The IF condition is necessary to prevent the insertion of the gateway's ID in the last position of the *polling_order* array.

Algorithm 4.1 The post order tree traversal for non binary trees algorithm, employed by the FM-WiFIX Scheduling mechanism running in the gateway.

```
 1: function POST_ORDER(node_id)
 2:     $node\_id\_descendants\_list = \text{GET\_DESCENDANTS\_LIST}(node\_id)$
 3:     for all $node_i \in node\_id\_descendants\_list$ do
 4:         POST_ORDER($node_i$)
 5:     end for
 6:     if $node\_id \neq gateway\_id$ then
 7:         Add node_id to the gateway's polling_order array.
 8:         Increment the iterator associated to this array.
 9:     end if
10: end function
```

Whenever a registration message is received, we recalculate the polling order and update the *polling_order* array. When choosing a new node to poll, the scheduling mechanism only has to pick the next node in this ordered array, as in PACE.

4.2.3 Topology Refresh

Under PACE operation, the gateway starts sending polling messages as soon as a node is registered. The topology refresh messages, broadcast periodically by the gateway, ensure that the network is constantly up to date, that new nodes are registered and polled and that dead nodes are removed from the list of nodes to poll. This process takes place simultaneously with the data exchange between the gateway and the nodes currently registered.

Under FM-WiFIX operation, we cannot use this scheme to ensure the topology refresh. At each moment, a large part of the network's nodes will have their Wi-Fi off, thus not being able to receive or respond to the topology refresh messages. Consequently, once a node shuts down its wireless radio interface, it would be considered a dead MAP by the former PACE logic and removed from the polling list.

Considering our scenario of a network of surveillance cameras, we can assume that the network topology does not change for large periods of time. Therefore, to address the previously described problem, our solution consists in an initial setup period, where we let PACE run normally, along with the topology refresh mechanism. Assuming that we know the number of sensor nodes in the network, once that number of sensors is registered at the gateway, we turn on the FM-WiFIX scheduling control and stop both the PACE's polling and the topology refresh mechanism (the gateway stops broadcasting periodical hello messages and the nodes stop choosing a neighbor periodically, based on the received hello messages).

To deal with sensor failures, our solution also includes a backup mechanism. When a node does not respond during its transmission opportunity, the error counter for that node is incremented. If the node responds to the polling in the next round, the counter is reset. If not, the counter is incremented again. If a node does not answer for three consecutive rounds, we assume that the network topology has changed. In that case, the FM-WiFIX control stops and we return to PACE's normal operation, i.e., the RDS no longer control the system: all nodes keep their Wi-Fi radio interfaces on and the polling messages are exchanged in-band, through Wi-Fi. This allows the reconfiguration of the network topology, without interrupting the video streams. The topology update ends when one of the following events occurs: the expected number of sensor nodes are all registered again at the gateway or after the expiration of a timeout. This timeout is set each time a new register message arrives at the gateway. In case of a node failure, the first condition would never be verified so, if the topology does not change during a large interval of time, we consider that the network is stable again and the system switches back to FM-WiFIX control. This feature also prevents a system failure if there is a sudden drop in the quality of the FM-RDS signal.

It is important to highlight that the described mechanism for starting the FM-WiFIX scheduling control is slightly different from the one described in the original proposal. The original proposal stated that the registration mechanism was executed periodically to deal with node registration failures and topology changes [2]. In our implementation, we already have the register acknowledgment message to deal with registration failures and we have a mechanism to detect failures. Thus, we only execute the registration mechanism at the initial setup and in case of node failures.

Another change between the original algorithm and the algorithm that we implemented is the Wi-Fi state at the beginning of the FM control. In the FM-WiFIX paper, it was assumed that after their successful registration, all nodes would switch off their Wi-Fi cards, except the gateway. In our solution, we use the PACE's registration mechanism (with a few modifications), which requires that all nodes have their Wi-Fi on, until the FM control starts and the first RDS polling message is received, as shown in Fig. 4.1a.

4.2.4 Data Exchange Between MAPs

A practical problem that arose during the FM-WiFIX's implementation was the fact that the PACE program was not prepared for the exchange of data generated by MAPs. In PACE's context, the MAPs' function is to provide Internet access to terminals, not generate data to send to the gateway. Therefore, when the gateway received data frames from a sensor node, as the source MAC address was from a MAP and not from a terminal, the gateway would interpret those frames as dummy polling responses, the kind of message that a MAP sends to the gateway whenever it doesn't have terminal data to transmit (in PACE's original implementation). Consequently, instead of deliver those frames to the application, the gateway would discard all the data received from the nodes, unless we were using an external IP camera, connected to the Raspberry Pi. However, what we were using was the camera module for the Raspberry Pi, connected through the Camera Serial Interface (CSI) port. This problem also thwarts the conduction of experiments using network analyzers such as Iperf or MGEN.

We tried to solve the problem by using virtual interfaces and the dummy interface module. None of these approaches were successful, apparently due to bugs in the Raspbian operating system. A similar approach was followed when testing the PACE implementation, with Access Points running OpenWrt, and using virtual interfaces to simulate terminals inside the MAP worked well.

As a last resort solution, we performed minor changes in PACE's code, in order to forward to the application all the frames received and not only the frames with a source MAC address from a terminal. As a consequence of this tweak, sometimes we would deliver dummy data to the application, specially on the sensor nodes side. To avoid this wrong practice, only packets larger than a dummy polling message (28 B) are delivered to the learning bridge.

4.2.5 FM-WiFIX Logic

Being described all the modifications done to the existent implementation of PACE-WiFIX, now we explain how the FM-WiFIX logic was implemented.

4.2.5.1 Gateway Side

The main gateway's responsibilities are: assigning ID's to the registered nodes; starting and stopping the FM control; scheduling the polling; and sending the RDS control messages.

After the registration phase, already described in Section 4.2.1, the FM control is turned on. This means that from now on, the PACE's polling logic will be override by the FM-WiFIX scheduling mechanism. This mechanism is mostly similar to the implemented by PACE. Upon reception of the data transmission from the node polled, we choose a new node to poll. This choice is as easy as picking the next element of the *polling_order* array, which was previously organized, during the registration phase. Knowing the ID of the next node to poll, we check if there is data to send to that sensor. With the ID of the next node to poll and a flag indicating if that node has data to receive or not, we generate the control data. The control protocol will be explained later, in 4.3.1, but the control data is, basically, a combination of the node's ID and the *hasData* flag. If the node has data to receive, after the generation of the RDS message the gateway will send a polling message with data address for that node, as would normally happen in PACE. If the node has no data to receive, the reception of the RDS message is already interpreted as the polling request and the node would immediately transmit. After the generation of the RDS message, the polling timeout is activated. If the polling timeout expires without the reception of a frame from the polled node, this exact process takes place, with the exception that we will count a failed polling attempt for that node, in order to identify a possible change in the network topology.

If one of the error counters reaches the limit of three (default number) consecutive failed polling attempts, a special control message ($ID = 0$) is generated, instructing the sensors to return to the PACE's polling mechanism. This special message is broadcast while the FM control is stopped, i.e., during the registration phase and during the network reconfiguration.

For testing purposes, there is also a special ID ($ID = 1000$) for terminating the program in all sensor nodes when the gateway program ends.

4.2.5.2 Sensor Side

The main responsibilities of a sensor node are: choosing a parent; performing its registration at the gateway; listen the RDS control messages and react accordingly to the control data received.

At startup, a sensor is operating under PACE's logic. It will choose a neighbor, send a registration message and snoop the registration messages that it forwards downwards, in order to know who their children are. During this phase, it will also respond to the polling messages received by returning polling messages with data or dummy polling messages, if it has no data to transmit upwards.

However, the program will also be listening the RDS messages broadcast by the gateway. While the gateway does not start the FM control, the message being broadcast is the one which stops the FM control. As soon as a RDS message is received containing a different and possible ID, the sensor node realizes that the FM control has begun. This is important for stopping the periodically topology refresh, where the sensor node chooses a new parent based on the hello messages received.

Every time a new control data is received, the sensor node parses the control data, acquiring two parameters: the ID of the node being polled and the *hasData* flag. After verifying if the values make sense, the node will decide the proper action to take, accordingly to the control data received.

If the RDS polling message is addressed for the node's ID and the *hasData* flag isn't set, the node turns on its Wi-Fi radio interface, a dummy polling message is injected in this sensor, in order to take advantage of the original PACE's mechanisms and, following this injected frame, the node will respond by sending its data (or a dummy polling return if it has no data to send). If the *hasData* flag is set, the node will just turn on its Wi-Fi and wait for the reception of a polling message with data sent by the gateway, exactly as would happen under PACE control.

If the RDS polling message is addressed for one of its descendants, the node's behavior will be similar to when it has data to receive. It will just turn the Wi-Fi on, wait for a frame reception and then, using the PACE's mechanisms, forward the received frame downwards.

When the RDS control message is not addressed either to itself or to one of its descendants, the node will turn its Wi-Fi off or keep with its Wi-Fi off and won't do anything.

Regarding the special control messages, if the RDS control has the ID 0, the node will leave the FM control state, returning to the normal operation of PACE and the topology refresh mechanism, which means that the node will periodically choose a neighbor again. If the RDS control contains the ID 1000, the FM-WiFIX program terminates.

Algorithm 4.2 summarizes the previous explanation.

4.3 The RDS Control Channel

The design and implementation of the RDS control protocol was highly influenced by the design and characteristics of the hardware and software used and also by the requirements of the RDS standard, presented in Section 3.4. Therefore, in this section, in addition to presenting the definition of the RDS control protocol, its implementation and integration, we also explain the operation and design of the software and hardware used, which served as the starting point for our implementation.

All the software used in the development of this RDS control channel was released under the GNU General Public License (GPL) (versions 2 and 3), giving us the freedom for changing the software or using parts of it into new programs [38].

4.3.1 The Control Protocol

The control protocol that enables the previously explained scheduling mechanism is extremely simple. It only consists on 9 bits: 1 bit flag that indicates if a sensor node has data to receive and an 8 bit identifier, as explained in Section 4.1.

Algorithm 4.2 The FM-WiFIX logic at the sensor side. This function is called every time a new control message is received and tells the node how to proceed accordingly.

```
1: function FM_WIFIX_LOGIC(rds_message)
2:     PARSE_CONTROL($rds\_message, \&id, \&hasData$)
3:     if $id == my\_id$ then
4:         Turn Wi-Fi on
5:         if hasData then
6:             Wait for frame reception (RECEIVE state)
7:         else
8:             Inject dummy polling frame (TRANSMIT state)
9:         end if
10:    else if $id \in my\_descendants\_list$ then
11:        Turn Wi-Fi on
12:        Wait for frame reception (FORWARD state)
13:    else if $id == STOP\_FM$ then
14:        Turn Wi-Fi on
15:        Stop FM control
16:    else if $id == TERMINATE$ then
17:        exit(0)
18:    else
19:        Turn Wi-Fi off (IDLE state)
20:    end if
21: end function
```

4.3.1.1 Fault Tolerance

By operating in the FM band, the control channel is affected by high noise levels. Thus, it is important to apply appropriate mechanisms of fault tolerance in our system, to assure that the system works well or, at least, degrades gracefully, in the presence of high error rates.

The new scheduling mechanism proposed requires that every sensor node receives the RDS control data. If that does not happen, a transmission or an energy saving opportunity will be wasted. A high presence of errors would significantly deteriorate both the network performance and the energy savings. Thereby, some fault tolerance strategies were considered, in order to achieve a more resilient system.

As explained in [39, Chapter 8], the key technique for achieving fault tolerance is to use redundancy. In our context, fault tolerance means that all nodes correctly receive the control data sent, even if errors are introduced in the RDS control channel. Three kinds of redundancy are possible: information redundancy, time redundancy and physical redundancy.

With **information redundancy**, extra bits are added to allow the recovery of corrupted bits. This is partially done by the RDS standard, as the checkword of each RDS block is sometimes capable of correcting corrupted data in the decoding procedure. By adding Forward Error Correction (FEC) combined with Error-correcting codes (EEC) such as Hamming codes, the error correction capabilities of the system would be further increased. However, this would be a completely non standard way of using the Radio Data System and would require much more work to implement

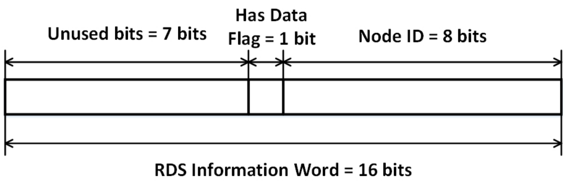


Figure 4.5: The information word format of each RDS block in our control protocol. There are 7 unused bits that could be used in future improvements to the control protocol.

the RDS transmitter and the RDS receivers. Moreover, due to the limited size of each RDS group and the RDS data rate, using strong EECs would not be feasible. Therefore, we adopted the simplest information redundancy technique: repeat the 9 bit control data in the four RDS blocks that compose the RDS message, as depicted in Fig. 4.6. If a message is received with errors, there are good chances that at least one of the blocks has the correct information.

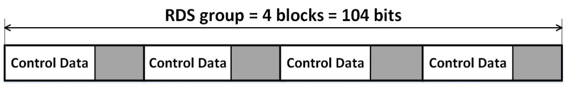


Figure 4.6: Redundant RDS message. Each block contains the same control data. The format of the control data field is shown in Fig. 4.5. The gray area corresponds to the check bits of each block.

With **time redundancy**, an action is performed and then, if needed, it is performed again. In our case, since the transmission of a RDS message is an idempotent action (i.e. can be applied multiple times without changing the result beyond the initial application), we can send the same message multiple times, to ensure its successful delivery to all nodes. Actually, this strategy is already employed by the RDS standard, which lists the recommended repetition rates for each message type [36, p. 18]. For our system, we empirically found the best compromise between error rate and rapid variation of the messages content. Since the estimated amount of time that a RDS messages takes to be generated is 88 ms (Eq. (4.1)), we only change the control data being transmitted every ≈154 ms, which should ensure enough time for the transmission of a complete RDS message.

$$RDS\_generation\_time = \frac{1}{\frac{RDS\_bit\_rate}{RDS\_message\_size}} = \frac{1}{\frac{1187.5\,\text{bit/s}}{104\,\text{bit}}} \approx 87.58\,\text{ms} \tag{4.1}$$

As previously explained in Section 3.4, the Radio Data System consists on a constant and, preferably, uninterrupted data stream. Thus, we cannot know for sure when the generation of a RDS message starts and ends, which explains the guard interval of 154 ms, to assure that, at least, one complete RDS message is sent each time. Increasing the number of RDS messages transmitted with the same information would increase the channel reliability but would also limit the rate at which we can poll the video sensors.

With **physical redundancy**, extra equipment or processes are added, to make it possible for the system, as a whole, to tolerate the loss or malfunctioning of some components. We have considered, as future work, study the inclusion of more gateways in the WVSN but, for now, we will only use one. Consequently, this kind of redundancy is not very helpful for our problem. Besides, the problem to solve here is overcoming the challenges introduced by the use of a noisy channel and not tolerating components faults.

4.3.2 RDS Integration

4.3.2.1 The RDS transmitter

Using only the Raspberry Pi hardware and a program called "Pi-FM-RDS" available in [40], it is possible to turn the Raspberry Pi into a FM-RDS transmitter. The program uses the Raspberry Pi's Pulse-Width Modulation (PWM) generator to produce Very High Frequency (VHF) signals. The radio frequency signal is emitted on GPIO 4, pin 7 on header P1 (see Fig. 5.1b).

The role of the Pi-FM-RDS program is to convert the RDS messages that we want to transmit into baseband samples, that will be used by the PWM generator to generate the FM signal. As already explained in Section 3.4, the RDS standard requires a precise bit rate of $1187.5\,\text{bit/s} \pm 0.125\,\text{bit/s}$. To efficiently assure that the baseband samples are sent at this exact rate to the PWM generator, the program relies on a Direct Memory Access (DMA) engine. The DMA is a hardware mechanism that allows peripheral components to transfer their I/O data directly to and from main memory, without going through the CPU [41, chapter 15]. Here, the program uses a DMA engine which has in its buffer the baseband samples and is in charge of sending them, at an exact rate, to the PWM generator. While the DMA is feeding the generator with samples, the program is mostly idle, thus saving significant amounts of processing power. The program just has to periodically wake in order to replenish the DMA buffer.

This process occurs as follows: samples are sent by the DMA at 228k samples per second. 4 samples correspond to a period of the carrier wave (carrier at 57 kHz [36, p. 6]). The basic clock frequency is obtained by dividing the transmitted carrier frequency by 48: $57k/48 = 1187.5$ [36, p. 8]. Therefore, 48 periods correspond to 1 bit. This logic is all assured by the DMA, the program just have to initialize the DMA engine and, then, periodically wake to replenish the DMA buffer. The rest of the time the program may be sleeping. Originally, the program awakes every 5 ms to generate more samples and passing them to the DMA buffer. In order to increase the polling speed, we have decreased that sleep time to nearly 149 µs. This allows the generation of a different RDS message every $\approx$154 ms.

4.3.2.2 Integration of Pi-FM-RDS with FM-WiFIX

FM-WiFIX must be able to pass to Pi-FM-RDS the control data that it wants to broadcast. In order to do so, we have to use some kind of Inter-Process Communication (IPC). Our choice must be compliant with some critical aspects of Pi-FM-RDS, which is a very delicate program due to the initialization of the DMA controller using memory owned by a user process. Pi-FM-RDS must ensure that the DMA engine is killed when the program terminates to avoid memory corruption [40]. Thus, as a preventive measure, any received POSIX signal triggers the program's termination. Moreover, Pi-FM-RDS must be continuously generating baseband samples and passing them to the DMA buffer, in order to avoid the interruption of the RDS data stream. Having analyzed the various options for IPC, presented in [42], we opted for implementing shared memory in the communications from FM-WiFIX to Pi-FM-RDS. Whenever the gateway wishes to send a new RDS messages, it writes on the shared memory segment the control data generated. On its side, when its time to start generating a new RDS message, Pi-FM-RDS sets the four RDS blocks with the data currently stored in the shared memory. This mechanism introduces some overhead but it is necessary for ensuring low error rates on the RDS control.

There is a case where the gateway must be informed that the RDS message was, with a high probability, correctly sent. This is the scenario where the polled sensor has data to receive. Upon reception of the RDS control message, the sensor must turn its Wi-Fi on and wait for the polling message with data, which implies that the gateway knows when the node will be waiting for its data and only sends the polling message at that time. This also requires some kind of IPC from Pi-FM-RDS to FM-WiFIX. However now we have different requirements that do not allow the reutilization of the shared memory segment. Herein, the acknowledgment from Pi-FM-RDS must interrupt the period where the gateway is waiting for data reception in the physical interface, without generating an interruption on the Pi-FM-RDS side. POSIX sockets are a suitable form of IPC for this scenario.

To sum up, when the gateway wants to transmit a new polling message, it writes the control data on the shared memory segment. The Pi-FM-RDS program will eventually start to transmit that data when it is time for a new RDS message to be generated. When the transmitter program generates a new RDS message, it is assumed that the previous RDS messages was successfully sent. Thus, at this point an acknowledgment is sent to FM-WiFIX through the socket connection. If the RDS message sent had the *hasData* flag set, this acknowledgment triggers the transmission of a polling message with data for the node being polled. Fig. 4.7 shows the interaction between the two processes. Note that a RDS message will probably be received at the sensor node before Pi-FM-RDS sends a confirmation to the gateway. This is because, to assure the correct dispatch of at least one correct RDS message, we only vary the message's content every 154 ms, when the generation time of a RDS message is $\approx$88 ms (Eq. (4.1)). If the message is correctly generated at the beginning of this large interval, the node polled will send its data sooner than the Pi-FM-RDS confirmation. When the gateway is waiting for the acknowledgment to send a polling with data, there will be an idle period. However, this is preferred to the interruption of the RDS data stream, which would cause a synchronization loss at the receivers.

4.3.2.3 The RDS Receiver

Initially, we were thinking about changing completely the standard format of RDS messages. In that case, using GNU Radio [43] for the RDS reception would simplify the task of parsing the RDS data stream into separate messages. Therefore, in the beginning we were using GNU Radio

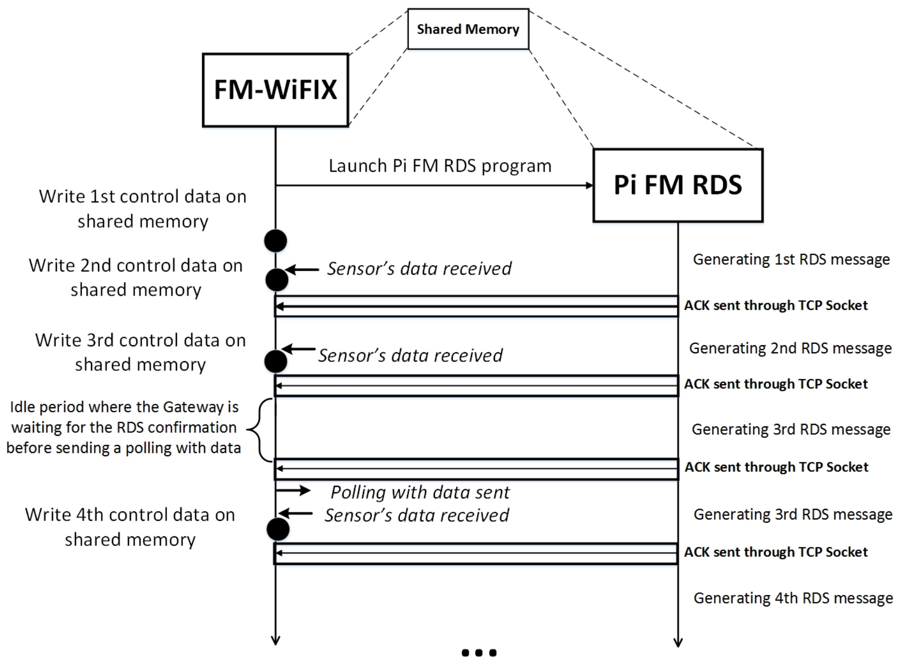

Shared Memory
FM-WiFIX
Pi FM RDS
Launch Pi FM RDS program
Write 1st control data on shared memory
Write 2nd control data on shared memory
Sensor's data received
Generating 1st RDS message
ACK sent through TCP Socket
Write 3rd control data on shared memory
Sensor's data received
Generating 2nd RDS message
ACK sent through TCP Socket
Idle period where the Gateway is waiting for the RDS confirmation before sending a polling with data
Generating 3rd RDS message
ACK sent through TCP Socket
Polling with data sent
Sensor's data received
Write 4th control data on shared memory
Generating 3rd RDS message
ACK sent through TCP Socket
Generating 4th RDS message

Figure 4.7: The communication between FM-WiFIX and Pi-FM-RDS at the **gateway**.

for the RDS reception, jointly with a DVB-T dongle. The installation and configuration of GNU Radio in the Raspberry Pi was not an easy task. When it was finally operational, we realized that the Raspberry Pi model B was not powerful enough to run GNU Radio. The lack of hardware requirements for running GNU Radio and the decision of adapting the standard block structure of RDS messages motivated us to abandon the GNU Radio path and choose another way of receiving the RDS emissions.

The final solution was a combination of the SparkFun FM Tuner Evaluation Board - Si4703 [44] and the program RdSpi [45], a Si4703 based RDS scanner for the Raspberry Pi that uses I^2C to communicate with the FM tuner and its registers, parses the RDS data stream and displays the messages received.

Several modifications have been performed to RdSpi. We modified this program to parse, uninterruptedly, the RDS data stream being received. The program now keeps track of the last control data received so that, when a RDS message is received, RdSpi is able to distinguish a new message from an old one. It checks all four blocks of each RDS message and interprets their content to confirm that the control data makes sense. Despite the 10 checkbits of each RDS block, we verified that sometimes, non-sense values were being received and passed to FM-WiFIX. If the received message contains new and valid control data, RdSpi must interrupt the waiting period of the FM-WiFIX program, running in the sensor node, to simulate the reception of a polling message. Once again, this requires, some kind of IPC. In this situation, there are no special requirements, it just must be something that interrupts the blocking period when the sensor node is waiting for data reception on the physical interface. The current PACE's implementation has a built-in Telnet server, that allows controlling the program in runtime. Issuing a Telnet control causes an interruption of the *select* function, where the program stops, waiting for input (whether from the wireless interface or from the Telnet server). For convenience, we used this system in the communication between RdSpi and FM-WiFIX, at the sensor nodes, as depicted in Fig. 4.8. Using a socket would introduce less overhead. However, issuing a Telnet command to *localhost* is also a fast operation, with negligible overhead when compared to the generation times of RDS messages.

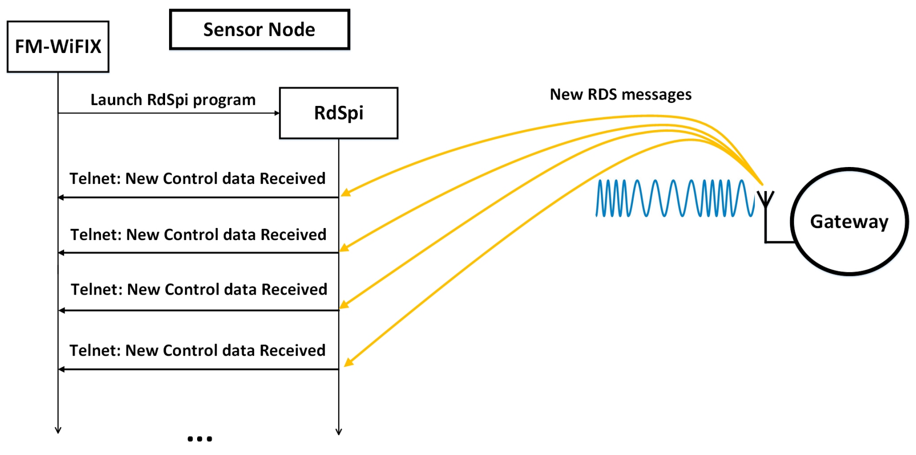

Sensor Node
FM-WiFIX
Launch RdSpi program
RdSpi
New RDS messages
Telnet: New Control data Received
Telnet: New Control data Received
Telnet: New Control data Received
Telnet: New Control data Received
Gateway
...

Figure 4.8: The communication between FM-WiFIX and RdSpi at the **sensor nodes**. When RdSpi detects that new and valid control data has been received, it sends that control data to the FM-WiFIX program, using the built-in Telnet server.

4.4 Improvement of the FM-WiFIX Solution to Tackle the Performance Issue

4.4.1 The Performance Issue

Both the modified solution being presented and the initial FM-WiFIX proposal suffer from one major problem: a massive loss in the network performance, when compared to PACE. One of the FM-WiFIX objectives was to keep the performance enabled by PACE. However, the time required for broadcasting a RDS message with an acceptable error rate is the system's bottleneck and decreases the maximum throughput to unacceptable levels. If we can only send a different RDS message every 154 ms, then we can poll, at the best, only $\approx$6.5 nodes per second. This limits the data reception at the gateway to only $\approx$6.5 packets/s which is a terrible value.

Currently, we are mimicking the PACE's symmetrical polling mechanism, where the gateway sends a polling message (a dummy one or with data), the polled node responds with a message containing its data and, upon reception of a response from the node polled, the gateway chooses a new node to poll, and the process goes on (see Fig. 4.9).

Our problem is the gap between the response and the reception of the new RDS polling message by the system's sensor nodes. The generation of a new RDS message is triggered by the response of the node polled. The time interval, between this time instant and the successful broadcast of a new RDS message, will usually be 154 ms, the rate at which we vary the content of the RDS messages. Sometimes this gap may be even higher, when errors are introduced in the FM-RDS control channel.

This big overhead is a physical limitation of the RDS techonology. The maximum goodput available is 730.8 bit/s (Eq. (3.1)) and we cannot start the transmission of a RDS message as soon as we want, due to the constant data stream nature of the Radio Data System. In an attempt to accomplish this thesis' objective and the FM-WiFIX proposal of implementing a scheduling mechanism based on the FM RDS, instead of changing the technology used to implement the control channel, we chose to adapt the polling mechanism to the limitations discovered.

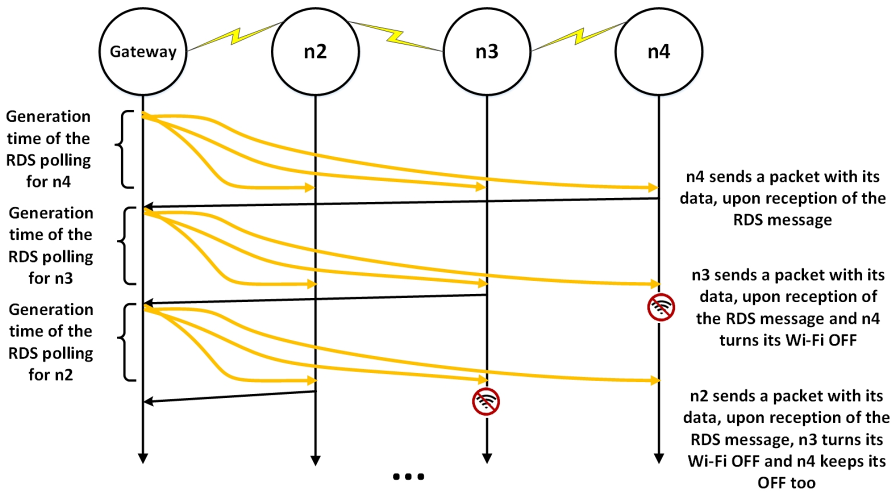

Gateway
n2
n3
n4
Generation time of the RDS polling for n4
Generation time of the RDS polling for n3
Generation time of the RDS polling for n2
n4 sends a packet with its data, upon reception of the RDS message
n3 sends a packet with its data, upon reception of the RDS message and n4 turns its Wi-Fi OFF
n2 sends a packet with its data, upon reception of the RDS message, n3 turns its Wi-Fi OFF and n4 keeps its OFF too
...

Figure 4.9: The PACE's polling logic adapted to our system. The generation time of the RDS messages is the bottleneck of the current solution.

4.4.2 The New and Final FM-WiFIX Solution

It is clear that the PACE's polling mechanism being employed is not suitable with the usage of FM-RDS. Therefore, we propose a new polling mechanism based on burst transmissions where, instead of transmitting one packet per polling opportunity, a node would continuously transmit, until the reception of a new RDS message or the expiration of a timeout. If the node has data to receive, then the node turns its Wi-Fi on, waits for the polling with data message from the gateway and only then starts its burst transmission. This new solution is depicted in Fig. 4.10 With this data burst mechanism, several aspects of the system are improved, at small and acceptable costs.

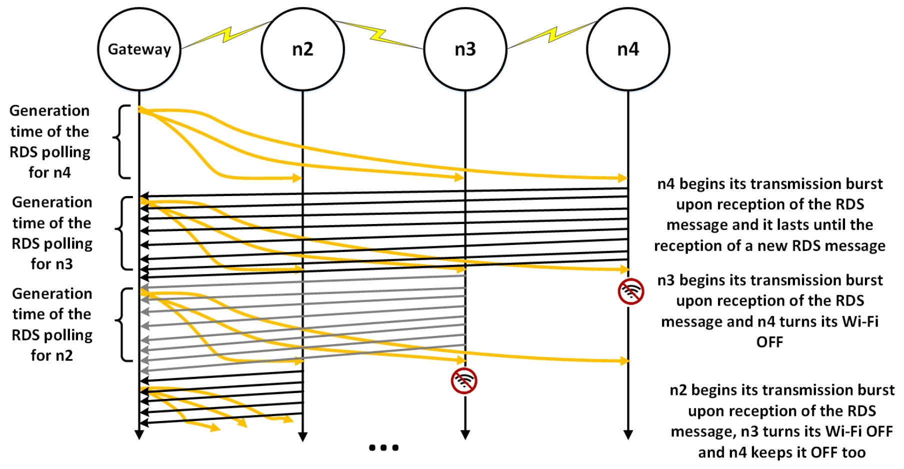

Gateway
n2
n3
n4
Generation time of the RDS polling for n4
Generation time of the RDS polling for n3
Generation time of the RDS polling for n2
n4 begins its transmission burst upon reception of the RDS message and it lasts until the reception of a new RDS message
n3 begins its transmission burst upon reception of the RDS message and n4 turns its Wi-Fi OFF
n2 begins its transmission burst upon reception of the RDS message, n3 turns its Wi-Fi OFF and n4 keeps it OFF too
...

Figure 4.10: The new polling logic employed. Now, for each polling opportunity assigned, a node starts a burst transmission, only stopping upon the reception of a new RDS message or the expiration of a timeout.

4.4.2.1 Paradigm Shift

The purpose of this improvement is the reestablishment of the network performance enabled by PACE, despite the physical limitations of the FM-RDS technology. The generation time of the RDS messages is now used for burst transmissions, instead of being wasted, thus **boosting the system's performance**, as desired.

The traffic pattern is now better adjusted to the Wireless Video Sensor Network (WVSN) scenario. This **new, ascendant traffic pattern** is preferred over the previous, symmetric one as the sensor nodes have much more data to transmit to the gateway than otherwise. A burst transmission for each polling message received fits perfectly into the WVSNs paradigm.

Regarding the network metrics, the system also suffered a significant improvement. Within a real-time video transmission scenario, the jitter and delay are important metrics to have in account. Our new polling mechanism **ensures low jitter values** due to the burst transmissions and also an almost **fixed delay between bursts**.

Finally, concerning the main purpose of this work, the energy savings, this solution should be quite similar to the previous one.

4.4.2.2 Challenges to Overcome

Despite all these improvements, this solution also introduces additional problems. The throughput fairness is slightly affected as, during the generation of the RDS messages, the nodes closer to the gateway will be able to transmit more packets, since their transmissions take less time to reach the sink node.

With the loss in symmetry and the control reduction, now errors in the FM-RDS control channel are critical and further measures must be employed to assure the system's reliability. If a node skips a control message, it will continue to transmit its data, which will be lost, either by the non-existence of a path to the gateway or by being discarded by it, as the gateway will no longer be expecting data from that node, that should be quiet and is transmitting. Moreover, nodes transmitting out of turn may cause collisions, which thwarts completely the PACE's philosophy that we aim to keep. To avoid the consequences of control related problems, we impose a maximum transmission time per polling received. This prevents the out-of-turn transmissions since, if a node does not receive the next RDS control message, it will soon stop transmitting anyway. The former identified problems of the node not turning on or not shutting its Wi-Fi off remain.

Finally, we must also have in account the rate at which a node should transmit packets during the burst. If a node performs its packet transmissions non-stop, without giving time for the packets to reach the gateway, it is probable that packets will be lost, due to collisions on the path to the gateway. As a node knows how far it is from the gateway, we addressed this problem by varying the waiting time between consecutive transmissions, depending on the hop count to reach the gateway. This is a linear function that was empirically validated and is defined as:

$$select\_timeout = 1000\,\mu\text{s} \cdot hop\_count \tag{4.2}$$

After each transmission, the sensor node stops at the *select* function, waiting for input, whether from the wireless interface or from the Telnet server. During the transmission burst, there will be nothing that interrupts this waiting period. Therefore, we control the burst speed by setting the *select* function timeout with the result of Eq. (4.2), as soon as the burst transmission starts.

When this timeout expires, if the node is in TRANSMIT state, a dummy polling frame is injected, triggering the transmission of a new packet to the gateway. If the node is in RECEIVE state, it will wait for the reception of a polling message with data and then will change to TRANSMIT state. The other states (FORWARD and IDLE) remain unchanged.

4.4.3 Alternative Implementation Path

Instead of controlling the data bursts with timeouts (between consecutive transmissions and for the burst maximum duration), we could simply use the already implemented PACE's polling logic. While waiting for the Pi FM RDS acknowledgment that the new RDS message was sent, the gateway could be exchanging polling requests and polling responses with the previous polled node, as much as possible until the RDS message is successfully sent. Upon reception of that acknowledgment, the gateway would start to generate the next RDS message, stop sending polling messages to the previous node polled and start to send polling messages to the node that was polled with the RDS message acknowledged. In this scenario, the sensor nodes would just turn their Wi-Fi on and off accordingly to the control messages and all the rest of the polling logic would be the same as in PACE (see Fig. 4.11).

This approach restores the original centralized and symmetrical control, as a node will only transmit upon reception of a polling request sent by the gateway, through Wi-Fi. However, it only overcomes the previous burst solution in the presence of many errors in the RDS control channel

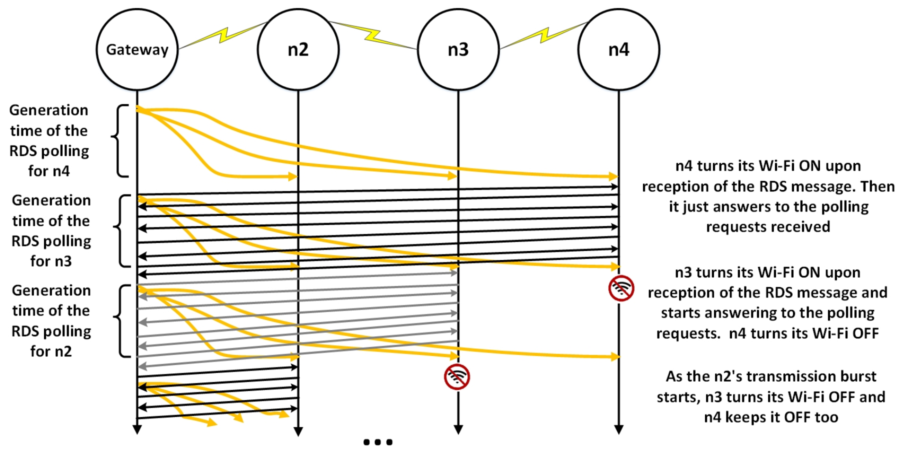

Gateway
n2
n3
n4
Generation time of the RDS polling for n4
Generation time of the RDS polling for n3
Generation time of the RDS polling for n2
n4 turns its Wi-Fi ON upon reception of the RDS message. Then it just answers to the polling requests received
n3 turns its Wi-Fi ON upon reception of the RDS message and starts answering to the polling requests. n4 turns its Wi-Fi OFF
As the n2's transmission burst starts, n3 turns its Wi-Fi OFF and n4 keeps it OFF too
•••

Figure 4.11: PACE's polling logic within transmission bursts. This solution is more centralized and more resilient to control errors but less suitable for the WVSN scenario, as it restores the symmetrical traffic pattern.

and in scenarios where all nodes are at a short hop count from the gateway. For nodes further from the gateway, the additional overhead of the polling request going downwards reduces the transmission opportunities that those nodes will have, thus reducing even more the throughput fairness, when compared with the previous burst solution.

This alternative implementation of the burst polling mechanism was not concluded since, overall, the initial burst solution with the timeouts well adjusted has more potential and is better suitable for a WVSN scenario. However, in some scenarios, the PACE original polling within data bursts could perform better and would be interesting, as future work, to implement both solutions, understand in which scenarios one is preferred to the other and maybe implement a dynamic solution where FM-WiFIX switches between the more suitable polling mechanism for each topology.

Chapter 5

Testbed and Results

The main goal of this thesis was to implement and evaluate the FM-WiFIX solution. Being the implementation process fully described in the last chapter, now we explain the methodology followed to carry out the experiments with the FM-WiFIX implementation. This includes the testbed setup, the choice of the most relevant metrics to evaluate, a description of the experiments carried out, and finally, the presentation and discussion of the obtained results.

5.1 Testbed Setup

In this section, we describe both the hardware and the software used in our testbed and the final assembling of all the components.

5.1.1 Raspberry Pi Model B

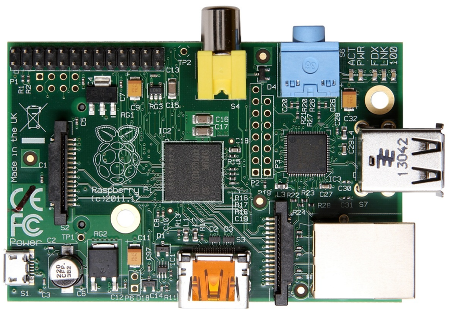
The Raspberry Pi model B was the platform proposed, in this thesis, to form our network of sensor nodes. This platform is characterized for its low price, reasonable processing power and being a very small computer, ideal to form a mesh network of small sensors.

The model that we will use in our work is the Raspberry Pi Model B, Revision 2.0 (Fig. 5.1). This model, like the other models from the first generation, has a Broadcom BCM2835 system on a chip (SoC) which integrates the following components [46, 47]:

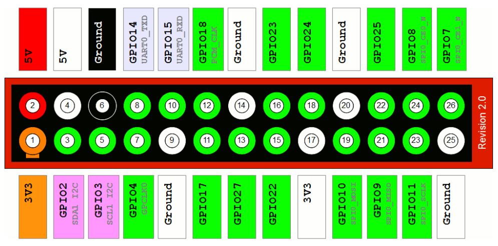
- CPU - Low Power 700 MHz ARM1176JZF-S single-core with ARMv6 CPU architecture
- GPU
  - Dual Core VideoCore IV® Multimedia Co-Processor
  - 1080p30 Full HD HP H.264 Video Encode/Decode
  - Low power, high performance OpenGL-ES® 1.1/2.0 VideoCore GPU. 1 Gigapixel per second fill rate
- SDRAM - 512 MB (shared with GPU)

- Digital Signal Processor (DSP)
- Two USB port

Besides the components integrated in the SoC, the model B has several inputs/outputs, being the most relevant for our work the two USB 2.0 ports (via the on-board 3-port USB hub) and a 10/100 Mbit/s Ethernet (8P8C) USB adapter on the third port of the USB hub. A relevant feature for our work are also the 8 General Purpose Input Output pins (GPIO), plus the following, which can also be used as GPIO pins: UART, I^{2}C bus, SPI bus with two chip selects, I^{2}S audio, +3.3 V, +5 V, ground [48] (Fig. 5.1b).



(a) The Raspberry Pi model B board.



(b) The Raspberry Pi model B GPIO Layout [48].

Figure 5.1: Raspberry Pi - Model B.

The operating system used was Raspbian [49], a Debian Wheezy armhf based operating system, optimized for the Raspberry Pi.

5.1.2 Wi-Fi Card

Since the Raspberry Pi does not come with a native Wi-Fi card, it was necessary to choose a compatible wireless adapter.

The first choice was the TP-LINK TL-WN823N [50] as this is a well-known dongle for its compatibility with the Raspberry Pi. However, at the beginning of our experiments, we realized that the driver available for this card in the Raspbian OS did not support the IBSS (ad hoc) mode.

Therefore, we equipped the Raspberries with the TL-WN722N dongle [51]. The Raspbian's driver for this Wi-Fi card, ath9k_htc, already supports ad hoc operation.

5.1.3 RDS generation and reception

5.1.3.1 RDS transmission

Regarding the RDS transmission, we have already explained, in Section 4.3.2.1, how the **Pi-FM-RDS program** [40] uses the Raspberry Pi's PWM generator to emit an FM-RDS signal through the GPIO 4. However, the strength of the signal, using the GPIO 4 alone, was too weak, allowing a range of only a few centimeters. To increase the transmitter's range, we crafted a simple antenna

(Fig. 5.2) with a 112 cm electrical wire and a connector, to couple with the Raspberry Pi's GPIO 4.

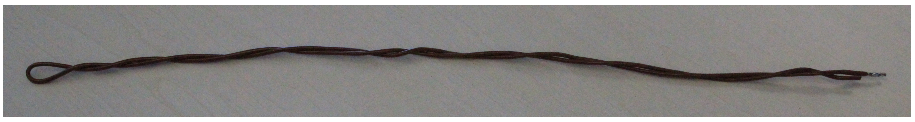
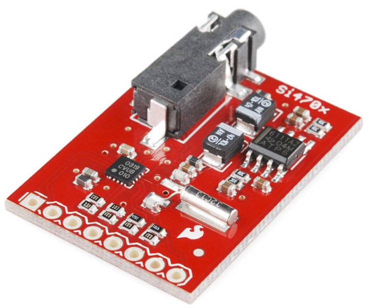


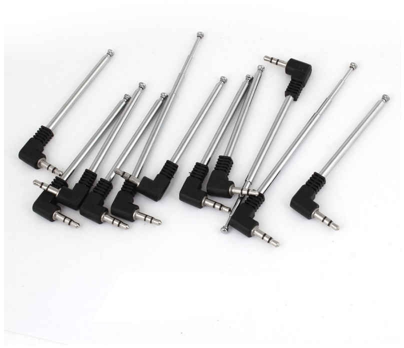
Figure 5.2: The antenna crafted to boost the transmitter's range.

5.1.3.2 RDS reception

For the FM-RDS reception, as previously stated in Section 4.3.2.3, we used the **SparkFun FM Tuner Evaluation Board - Si4703** [44]. This small and cheap FM radio, which is shown in Fig. 5.3a, is also capable of detecting and processing RDS information. After some experiments, we notice that the reception of our RDS transmissions, with the board alone, was very poor. Thus, we ordered a set of telescopic antennas to improve the RDS reception. These antennas have a 3.5 mm audio jack connector, a maximum diameter of 6 mm and a full length of 245 mm



(a) The Si4703 FM Tuner Evaluation board [44].



(b) The telescopic antennas used with the FM Tuner board, to improve the RDS reception.

Figure 5.3: The equipment used to enable the RDS reception

The **RdSpi program** [45] is in charge of this tuner's control. Using an I^2C connection, this program is able to interact with the Si4703 FM tuner chip's registers, as was referred in Section 4.3.2.3. The schematic presented in Fig. 5.4 shows how to connect the FM tuner board to the Raspberry Pi.

5.1.4 Raspberry Pi Camera Module

Finally, we needed to equip our sensor nodes with a video camera, in order to truly achieve a Wireless Video Sensor Network (WVSN). For this purpose, we used the Raspberry Pi camera module [52], which couples with the Raspberry board through the Camera Serial Interface (CSI) port.

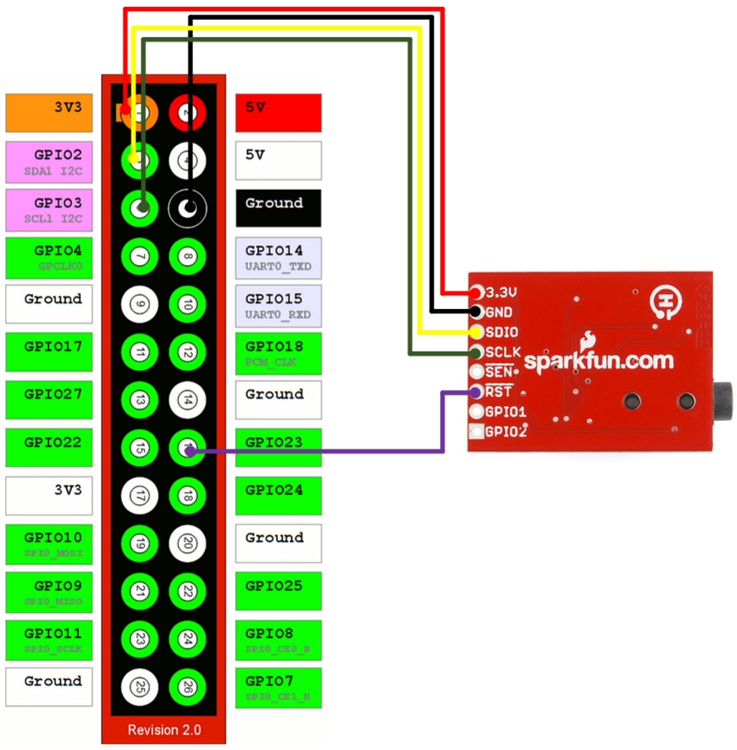


Figure 5.4: A schematic that shows how to correctly connect the SparkFun FM Tuner Evaluation Board - Si4703 to the Raspberry Pi [44, 48].

After enabling the Pi to work with the Raspberry Pi camera, in the *raspi-config* menu, the camera module is ready to use.

For generating the video stream at each sensor node, we used the command line tool for capturing video with the camera module, raspivid [53], together with cvlc, the console VLC player. The following command would start a RTSP stream [54]:

```
$ raspivid -o - -t 0 -n | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp
    ://:8554/}' :demux=h264
```

The cvlc player is also used at the gateway to receive the sensor's streams.

5.1.5 Testbed Assembly

To conclude our testbed, now we must put all things together, in order to achieve a functional network of video surveillance cameras.

We have assembled 6 video sensor nodes and 1 gateway node. Each node has a TL-WN722N wireless adapter and an Ethernet connection, to allow us to externally control the node during the experiments (e.g. running network analyzers, retrieving results, starting and stopping the tests). The sensor nodes are equipped with a Si4703 FM tuner board and a telescopic antenna while the gateway node has attached to its GPIO 4 the crafted antenna, to improve the FM broadcast range.

Regarding the Ethernet control network, all nodes are connected in the same VLAN and are controlled recurring to a laptop, external to the WVSN. To enable this connectivity, several switches were used, as can be seen in Fig. 5.5. Moreover, they are all connected in an ad hoc network through their wireless interface.

To automatically configure each Raspberry Pi in the two networks and also to deploy and build the developed programs over the network, we wrote some bash scripts, to automatize both the development and the testing phases.

The final testbed setup is shown in Fig. 5.5.

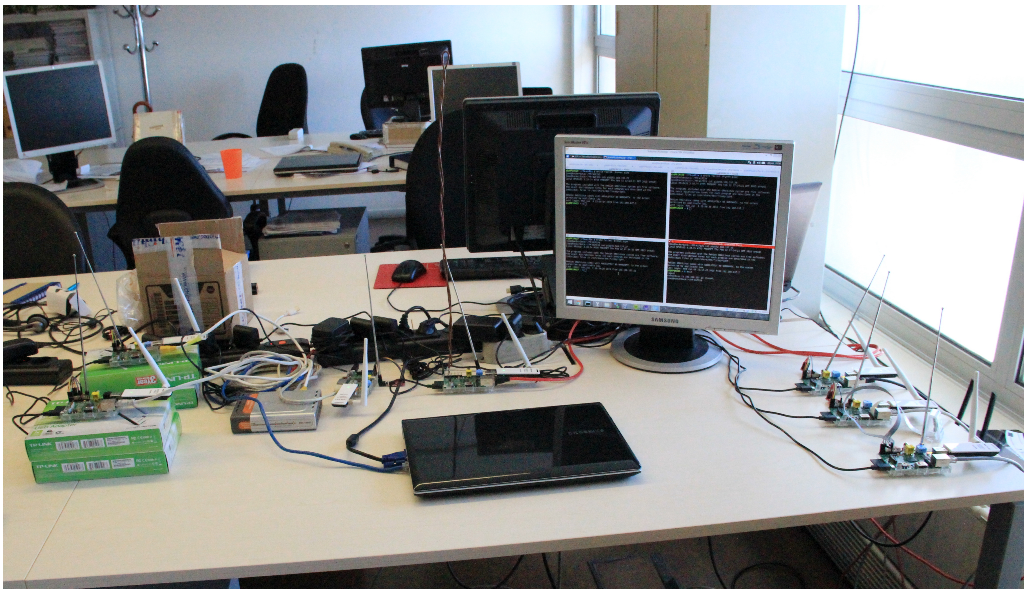


Figure 5.5: The final testbed.

5.2 Metrics to Evaluate

In this section we describe the metrics that we found more relevant to retrieve from our tests and to use in the evaluation of our system.

5.2.1 Energy Consumption

Measuring the energy consumption or the energy savings is the most important metric to retrieve, since the purpose of the FM-WiFIX solution is precisely to improve the network's energy efficiency.

For measuring this metric, our initial idea was to use a high precision multimeter, with a high sample rate, to calculate the total power consumption of the sensor node during its operation, while constantly switching the Wi-Fi on and off.

As we later explain, in Section 5.3, effectively shut down the Wi-Fi cards was not possible, so we had to use another approach to measure the energy consumption. We used the aforementioned multimeter to measure the Raspberry Pi consumptions with and without the Wi-Fi card and programmed the FM-WiFIX implementation to account the time that the node would be with its Wi-FI on and off, accordingly to the control messages received. Based on these calculations, we can have a very accurate idea of what the energy consumptions would really be, using the FM-WiFIX solution. This process and results are better explained in Section 5.3.

5.2.2 Throughput

Throughput is the quantity of error free data transmitted per unit of time (or the average rate of successful message delivery over a communication channel). Its value is always smaller than bandwidth (the data carrying capacity of the communication channel) and it could be measured in bit/s, byte/s or packets per second (pps).

This metric can be easily measured using Iperf [55], a well-known tool for network performance measurement. Iperf creates TCP and UDP traffic and then measures the throughput and delay jitter of the network carrying it. In the process, it allows the tuning of several parameters to achieve more accurate results.

5.2.3 Packet Delay Variation

Packet delay variation, also known as jitter, is another important metric to be measured. It represents the difference in the end-to-end delay between packets successfully received. This metric was proposed and defined in [56]. In a real-time video streaming application, it is important to assure low jitter values.

As previously stated, Iperf is also a suitable tool to measure the packet delay variation (jitter).

5.2.4 Error Ratio

The error ratio is another typical metric retrieved when evaluating a network's performance. We will also use Iperf to measure the error ratio but having in mind that sometimes, the error ratio measured by Iperf is inflated, since when its buffers are full, Iperf starts to discard packets. To identify this kind of situations, we also retrieve some statistics in the FM-WiFIX program that give us an estimation of the error ratio.

5.2.5 Fairness

Finally, we will verify if all the nodes are receiving the same transmission opportunities. We already know that, with the new polling mechanism, the fairness enabled by PACE will not be completely maintained. However, we expect to avoid big discrepancies. Using the throughput measurements of each node, we can easily check if there is fairness in the transmissions and, if not, which nodes are having troubles, in order to fix them.

5.3 Energy Saving Estimations

The goal of this thesis is to prove that the FM-WiFIX concept effectively has a significant impact in the reduction of the power consumption of a WVSN. If we had been capable of, effectively, turn on and off the Wi-Fi radio interfaces, we would measure the power consumption of the system during its operation time and could also make some tests with batteries, to understand how our optimization would affect the system's lifetime when the nodes are battery powered. As this was not accomplished neither was the main goal of this work, we were forced to follow an estimation approach. Nevertheless, we consider that the results obtained using this methodology are valuable and give us a very approximate idea of the real energy savings achieved by this solution, in a real operation scenario.

In this section, we firstly explain the challenge of effectively switching the Wi-Fi on and off and the research done for a suitable solution. Then, we explain how we measure the energy savings by estimation. Finally, we conclude this section by showing the obtained results, about the power consumptions of the Raspberry Pi and the associated hardware, used in this testbed. This results will be the starting point for calculating the energy consumption of our system in the multiple scenarios and experiments described in Section 5.5.

5.3.1 The Power Saving Challenge

Rapidly switching the Wi-Fi card between on and off states is far from being a straightforward task. If the sensor nodes only had to wake up sporadically or could take a few seconds (>4 s) to reconnect to the network, this would be an easy job. However, in our system, a node must be able to switch its Wi-Fi on and off within milliseconds, otherwise the scheduling mechanism would not be efficient. This is precisely the innovation and the challenge of the proposed FM-WiFIX solution, when compared to the other out-of-band solutions presented in the state of the art (Section 2.1). Therefore, this is an ambitious proposal with many challenges to overcome. The bottleneck introduced by the generation time of the RDS messages was one of them. This difficulty in effectively switching the Wi-Fi on and off is another.

Shutting the wireless interface down is relatively easy. By issuing the command *$ ifconfig wlan0 down*, we bring the interface down, which already consumes much less energy. To completely remove the Wi-Fi's energy slice from the equation, we could additionally use a switch to cut out the power to the USB port, where the Wi-Fi dongle is connected. The Raspberry Pi platform does not allow this power cut to a specific USB port but, we could use a USB hub such as the Yepkit USB Switchable Hub (YKUSH) [57], which implements a protocol to correctly shutting down a USB port and has good integration with Linux and bash commands (or we could just assemble an external switch circuit). This would allow to completely eliminate the Wi-Fi power consumption during idle periods.

The problem arises at the time of switching the Wi-Fi back on. Bringing the interface up and, mainly, the reconnection to the ad hoc network, are operations that take a few seconds to be completed. We tried two different paths, presented below, to overcome this problem.

5.3.1.1 Ad Hoc Fast Association

In order to achieve the original objective of effectively turning off the Wi-Fi radio interfaces, we tried to change the mechanism that governs the Wi-Fi association to an ad hoc network to allow a fast association to the network, when we bring the wireless interface up again. This would imply the use of the mac80211 module [58]. This module is a subsystem to the Linux kernel, which contains the Medium Access Control (MAC) sublayer management entity (MLME), the entity where the MAC state machines are located. Fig. 5.6 shows this module's location in the system's architecture. Our main interest in this module goes to the *ibss.c* file, where the IBSS MLME is implemented.

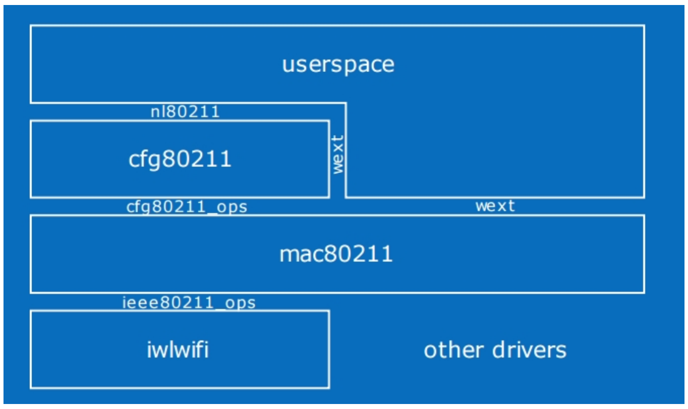


Figure 5.6: Location of the mac80211 module in the system's architecture [58].

Our first approach was based on the following mechanism. In the process of association with an ad hoc network, a device first checks if a network with the ESSID provided already exists. To do so, it scans the medium for a while, listening for beacons. If the network does not exist, then the node creates a new ad hoc network, after that scanning period. Based on some comments in *ibss.c*, a file from the source code of the mac80211 module [59], we found that, by issuing the following command, we were able to avoid this scan, thus decreasing the association time:

```
$ sudo iw dev wlan0 ibss join fmwifix-testbed 2417 fixed-freq 16:D5:2E:42:54:50
```

However, this still was not fast enough. Later, we realized that setting the interface to operate in ad hoc mode, without performing any association action, was also time consuming. We still tried to modify some extra files to force the default operation mode to be the IBSS mode, but with no success, all these actions were taking too long to be completed. Sometimes it would take just one second, other times it would take almost four to bring the interface up and associate with the ad hoc network. But even one second would be too much for the requirements of our scheduling mechanism. Consequently, we gave up on this path.

5.3.1.2 Forcing IBSS Sleep State

Being incapable of effectively shutting down the wireless interfaces, without compromising the system's operation, we tried the path of only forcing the card to enter into a power saving state. The energy savings would not be as good as the ones achieved if we effectively shut down the wireless interface but would still be significant, based on the differences shown in Table 2.1. Within this approach, our solution would bypass the IEEE 802.11 Power-saving mode protocol, that would keep the nodes always on, as they have always data to transmit, in our real-time video scenario. If we were successful, during idle operation, the node would switch to sleep state, where the power consumption is significantly reduced but the association with the ad hoc network is maintained, allowing a rapid wake up. With this solution, our out-of-band control channel would just schedule the IEEE 802.11 Power saving mode, replacing the ATIM frames.

We discovered three potential ways of achieving this control over the card, to force it to enter into sleep state: by issuing some commands implemented by the drivers, by using and adapting the Mesh power save [60], which is mainly implemented inside the mac80211 module, and by directly interacting with the chip registers, bypassing the driver and the mac80211 module. Regarding the drivers, we were faced with a serious lack of support for the power saving mode feature in ad hoc mode. None of the three chipsets tried (RT5370, RTL8192CU and AR9271) had a Linux driver with the Power saving mode correctly implemented in IBSS (ad hoc) mode. The driver that we used, ath9k_htc, has even the power saving mode for infrastructure mode disabled by default, due to its lack of reliability. Therefore, the only option left was to modify the driver used with our NIC for ourselves, which is a difficult and laborious task for the scope of this thesis. We still had a look at the ath9k_htc firmware source code but there was no easy way of modifying it to achieve this control over the card's sleep state.

In the implementation of similar solutions, where the authors tried to control the IEEE 802.11 PSM, they were only successful because they have made a custom driver [61, 5]. We conclude that, for the full implementation of the FM-WiFIX solution, another custom driver will be required, if we want to use an ordinary Wi-Fi dongle. However, as was demonstrated in [61], if the sleep state is hardware-implemented, then the transitions between sleep and idle states are very efficient, taking about 0.4 ms to be performed. Therefore, our approach of accounting the time that the node will be with its Wi-Fi on and off within the FM-WiFIX program, based on the control messages received, is quite realistic.

With the impossibility of forcing the sleep state in our Wi-Fi dongle, we have also conducted some research about Wi-Fi modules that would allow this kind of direct control over power saving states. One interesting candidate was the Wi-Fi Module ESP8266 [62], a cheap and versatile SoC which allows low level access, namely for power saving functionalities. However, this module does not support ad hoc operation, which is not acceptable in our context. There is only a mechanism where, each node, works simultaneously as AP and station, which allows simulating a mesh network, but it has some bugs and presents bad performances.

Similar to what we explain in Section 5.3.1.1, we could use Mesh power save [60], which is mainly implemented in the mac80211 module, to control these transitions between sleep and active state. The approach would be essentially the same as the one with the drivers (using our program to bypass the normal power saving mechanisms) and would also require too much work for the context of this thesis. This is, however, the best bet for reaching the goal of forcing the Wi-Fi cards into sleep state.

Finally, the option of directly interacting with the chip registers and forcing it to enter into sleep state. Although its potential, we are not certain that such method would still guarantee the association to the ad hoc network, during the sleep period, and the AR9271 chip's data sheet (the chipset of the Wi-Fi dongle used in our testbed) did not provide specific details about the registers that needed to be set to force this sleep state [63, p. 105].

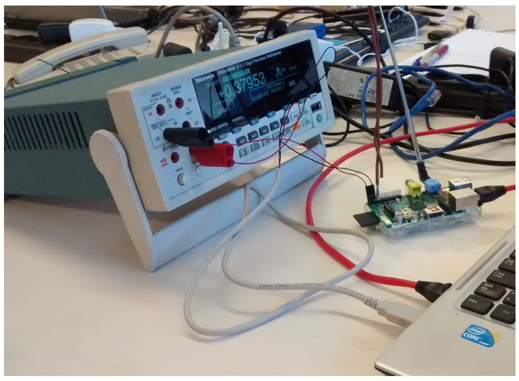
Due to these setbacks, we decided that it was best to abandon this path of forcing the Wi-Fi card into sleep state.

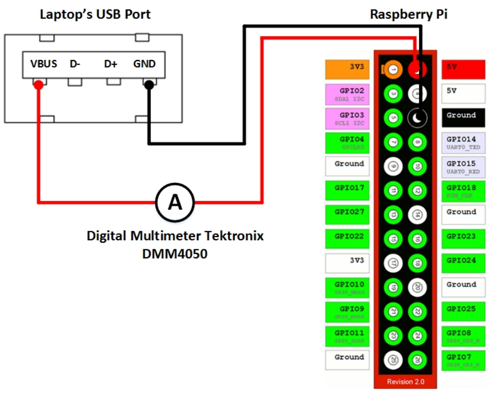
5.3.2 Power Measurements to the System's Components

The estimation of the energy gains, achieved by our solution, demands an initial study about the power consumption of the system's components, specifically the power consumption of the FM-RDS receiver and the Wi-Fi card, in idle state. Therefore, we now describe how we have conducted the experiments that allowed us to characterize the power consumptions of each one of the system's components and present the obtained results.

5.3.2.1 Experimental Procedure

To accurately measure the power consumptions associated to our system, we used the high precision digital multimeter Tektronix DMM4050 [64]. This multimeter was used as an ammeter, placed between the power source, a laptop's USB port, and the pin 2 of the Raspberry Pi, which corresponds to 5 V. This way we can precisely measure the electrical current being consumed by the Raspberry Pi, without the interference of the power supply usually employed to power this device. The measurement setup and the circuit diagram are shown in Fig. 5.7.


(a) The measurement setup.



(b) Circuit representation [65, 48].

Figure 5.7: Power measurement procedure.

The methodology employed for the measurements was the following. For each scenario, we took 1000 measures of the current being consumed by the Raspberry Pi, over a period of 42 s (sample rate of $\approx$ 24 samples per second). Then by calculating the average current value, $\overline{I}$, and knowing that the Pi is powered with 5 V, we use the electric power formula to find the power consumption:

$$P = V \cdot I \Leftrightarrow P = 5\,\mathrm{V} \cdot \overline{I} \quad (W) \tag{5.1}$$

5.3.2.2 Power consumptions

The consumption scenarios considered were the following:

1. Base consumption, i.e., the Raspberry Pi alone, with only an SSH session over Ethernet for control purposes;
2. Si4703 FM-Tuner, connected to the Raspberry Pi;
3. Receiving an FM-RDS emission;
4. Transmitting FM-RDS;
5. TL-WN722N Wi-Fi dongle, plugged into the Raspberry's USB port, with the wireless interface down;
6. Wireless interface up, in idle state;
7. Camera module attached. We performed four experiments with the camera, sending a stream, via Ethernet, to another device, with and without capturing movement, and only retrieving the video, with and without movement, but with no one receiving the stream. As the results obtained were approximately equal, we just present an average value of the four tests.

By subtracting the base consumption value to the measured value when the system is performing a specific action or powering an external device, we get the energy consumed by that action or device. Table 5.1 shows the power consumptions obtained for the above-mentioned scenarios. The same information is graphically represented in Fig. 5.8.

Based on these results, we have calculated the percentage of the energy consumed by each system's component in the overall power consumption of a sensor node, which, without the FM-WiFIX solution, would have a camera and a Wi-Fi dongle attached. By adding the Si4703 FM tuner to control the Wi-Fi dongle power state, we could save 21% of the total energy consumed by a sensor node in idle state, as shown in Fig. 5.9. We estimate that the total power consumption of a sensor node, in idle state, with no optimization, is $\approx$3114 mW. Therefore, based on the previous energy measurements, using the FM-WiFIX solution, when the node is in idle state, we save $0.21 \cdot 3114\,\mathrm{mW} \approx 654\,\mathrm{mW}$. From the perspective of extending the system's lifetime, please

Scenario	Power Consumption
1 - Base Consumption	1898 mW
2 - Si4703 FM tuner	99 mW
3 - FM-RDS Rx	106 mW
4 - FM-RDS Tx	195 mW
5 - Wi-Fi Dongle	381 mW
6 - Wi-Fi Idle State	750 mW
7 - Camera module	467 mW

Table 5.1: Power Consumptions of the different scenarios.

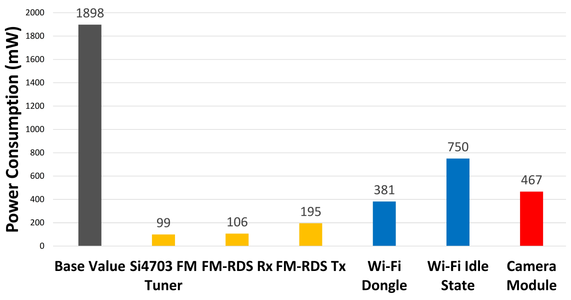
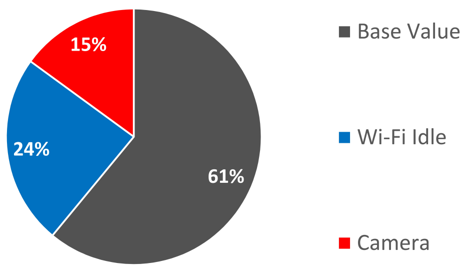
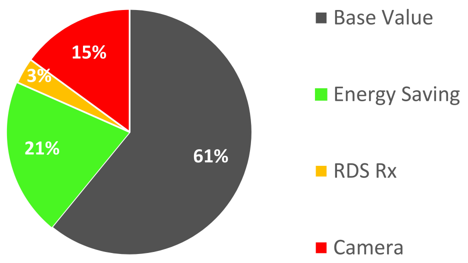


Figure 5.8: Comparison between power consumptions in different scenarios.

note that in a real system, using dedicated video sensors, the Wi-Fi card would consume a larger percentage of the total energy consumed, since a sensor of this kind is likely to be more energy efficient than a multi-purpose platform, as the Raspberry Pi.



(a) Energy consumed per hardware component.



(b) Energy savings in idle state, enabled by the out-of-band control channel.

Figure 5.9: Comparison between a sensor node's energy consumptions, in idle state, with and without the RDS control channel.

5.4 RDS Reliability and Variation Rate

As explained in Section 3.4 and Section 4.3, the RDS error ratio is related to the rate at which we vary the content of the RDS messages. If every RDS message correctly generated has different data, then we do not have time redundancy, a fault tolerance technique vastly employed in the RDS standard. To better characterize the cost of sending polling messages at faster rates, we have conducted the following experiments in an indoor scenario.

We have placed several RDS receivers inside a house, with a fixed RDS transmitter, and retrieved data about the amount of RDS messages lost by the receivers, which are at different distances from the gateway, in function of the rate at which we generate RDS messages with new content. All the receivers were placed at a height of ≈1 m. Each experiment was repeated four times, in order to guarantee the validity of the results.

To start, we conducted an experiment with the original programs used for transmitting and receiving RDS (Pi FM RDS and RdSpi). On this experiment, every point in the blueprint (Fig. 5.10) would receive the RDS broadcast with a negligible error ratio, as long as the receiver's antenna is placed at a height of at least 1 m. In these conditions, RDS achieves much better ranges than Wi-Fi, as can also be seen in Fig. 5.10.

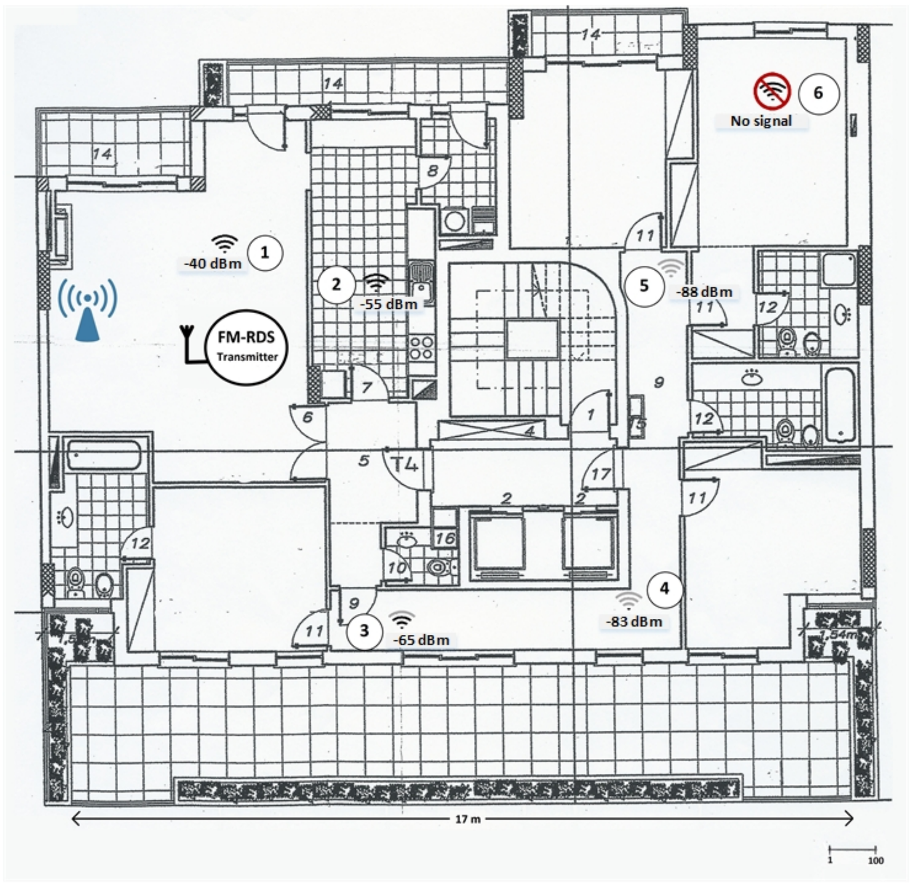


Figure 5.10: Location of the RDS receivers within an indoor scenario. The Wi-Fi signal strength was measured using the Wifi Analyzer app for Android [66].

However, we were only changing the content of the RDS messages every ≈5 s, which is obviously unacceptable for the requirements of our scheduling mechanism. As explained in Section 4.3, we reduce the generation time of new RDS messages from 5 s to 154 ms. At short distances from the gateway, this presents acceptable error ratios. Nevertheless, we find important to characterize the RDS control channel, regarding the error ratio, in function of the variation time of the content of RDS messages and the distance to the RDS transmitter.

The following results were obtained using the modified transmitter and receiver programs, that we use in our FM-WiFIX implementation. The most significant change is the interval between the generation of new RDS messages. We conducted these experiments in an attempt to find the best trade-off between range and error ratio, maintaining an acceptable variation time of the content of the RDS messages. Thus, we tried four acceptable values for this variation rate: 154 ms, 169 ms, 225 ms and 281 ms. Fig. 5.11 presents the obtained results, for every location of Fig. 5.10.

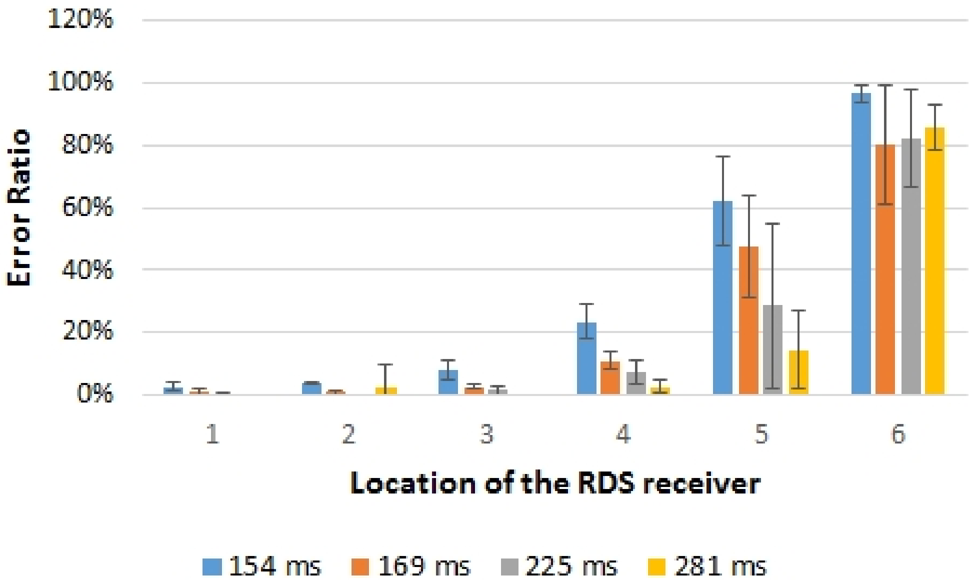


Figure 5.11: The error ratios for each generation rate of new RDS messages, for every point depicted in Fig. 5.10. A confidence interval of 95 % is also represented.

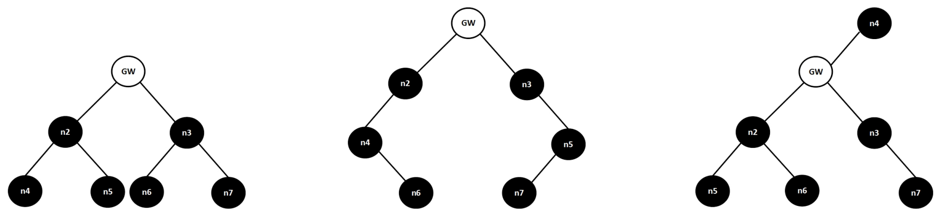
As expected, the error ratio increases with the distance and the obstacles between the transmitter and the receiver. As was also expected, if we give more time between generating RDS messages with new data, the error ratio usually decreases. However, the main conclusion that we draw from these experiments is that, for our proof-of-concept scenario, we would not achieve any significant improvement by increasing the previously defined interval of 154 ms. The distances between the receivers and the transmitter in our testbed are equivalent to the distance between the point number one and the RDS transmitter, in Fig. 5.10. Other tests showed that smaller intervals between new RDS messages would present high error ratios, even at close distances.

Concerning the reliability of the RDS channel at farther distances, we reinforce our conclusion that it brings several challenges, which can only be addressed by changing the scheduling protocol (e.g. transmission burst). Besides introducing significant delays between transmissions of a given node, it also compromises the range of the control channel, that must be large enough to cover the entire mesh network.

5.5 FM-WiFIX Evaluation

To conclude this work, we now proceed to the proof-of-concept of the FM-WiFIX solution and its respective evaluation. To proper evaluate our implementation, we use both the statistics accounted by our FM-WiFIX program and the network analyzer tool Iperf [55]. The experiments consist in using Iperf to generate traffic to the gateway, simultaneously from all nodes, as would happen in a real video scenario. At the gateway node, a bash script is in charge of starting an Iperf server for each client, in six different ports. Each sensor node will connect to one of those ports and send their data during 60 s. At the end of each experiment, the Iperf server calculates the statistics for each sensor and save them in a log file. There is also a bash script to retrieve all the statistics logs, generated by FM-WiFIX, at each node.

We will evaluate the FM-WiFIX solution in three different scenarios (Fig. 5.12) and compare its gains and performance to the ones achieved by PACE, in the same context. To force a desired network topology, we implemented a function on the FM-WiFIX program to block all frames from a given MAC address. Each node is identified by the last octet of its IP address. All pairs MAC-identifier are hard-coded in our program. Then, knowing the identifier of the node running the program and the desired topology, we will block all the non-neighbor MACs in the network. This means that a node will only receive hello messages from the node that is supposed to be its parent, thus allowing us to force any desired topology.



(a) Binary Tree Topology.
(b) A balanced tree topology with two branches and three hops.
(c) Unbalanced tree topology.

Figure 5.12: The three scenarios where the FM-WiFIX solution was evaluated.

5.5.1 Testbed Limitations

Shortly after the beginning of the first experiments, we realized that our testbed had some serious limitations. Based on the results presented in PACE proposal [11], a network with only one active node and at 1 hop from the gateway, should be able to achieve a goodput of nearly 30 Mbit/s. However, in the same conditions, running the original PACE program, one of our sensors could hardly achieve a throughput of 900 kbit/s. Our theory is that the complex and low level operations performed by WiFIX and PACE, regarding the system's network interfaces (e.g. the multiple transitions between kernel and user space, when a frame is being exchanged between the network interfaces, the WiFIX daemon and the NIC), are less efficient on the Raspberry Pi architecture (ARM) or on the Operating System used (Raspbian). In fact, we had several problems during the development of this thesis due to limitations of the Raspbian OS. The results presented in [11] were obtained using Access Points with a MIPS architecture and the OpenWrt OS, which is a quite different setup. Moreover, when the gateway is running the FM-WiFIX program, the Pi FM RDS program and six Iperf servers, the Raspberry Pi's processor operates at 100 % of its capacity. Therefore, we conclude that the Raspberry Pi platform is not a suitable solution for implementing FM-WiFIX and that it is the cause of the low throughput values obtained. However, as we test the original PACE program in the same conditions, and this lack of efficiency only affects the network performance and not the accounting of the energy savings, **our results and conclusions remain valid**.

5.5.2 Results

The obtained results are now presented. In wireless communications, we deal with stochastic processes, our experiments are not deterministic. Therefore, we must repeat the same experiment several times, in order to assure a satisfactory confidence interval. To do so, we retrieved our results by repeating the previously described experimental procedure 8 times, for both solutions, in each scenario. Due to the testbed limitations aforementioned, we limit the traffic generated by Iperf to only 350 kbit/s, which is enough to saturate the network. All the error bounds depicted represent a confidence interval of 95 %, obtained using the Student's t-distribution due to the small sizes of our data sets (we had to discard the results of some experiments due to Iperf failures, certainly caused by the lack of processing power at the gateway node).

The **error ratio** measured by Iperf in the three scenarios was negligible for both solutions.

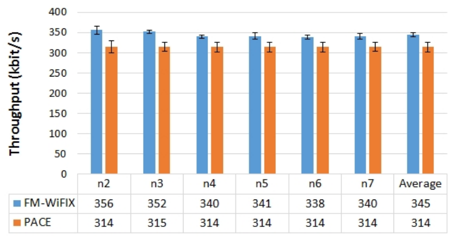
5.5.2.1 Binary Tree Topology

The graphs presented by Fig. 5.13 shows the results obtained, per node, for the binary tree topology (the topologies are represented in Fig. 5.12).

Fig. 5.13a shows that the fairness provided by PACE was maintained in our solution and, thanks to the new scheduling mechanism based on transmission bursts, we were even able to slightly improve the PACE's performance, while saving significant amounts of energy by leaving the Wi-Fi wireless interfaces off during the majority of the operation time (see Fig. 5.13c and Fig. 5.13d). However, this comes at the cost of increased jitter and delay.

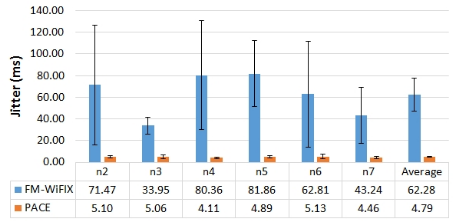
5.5.2.2 Three Hop, Balanced Tree Topology

This is a more demanding scenario, due to its two branches with three hops (see Fig. 5.12b). However, our solution outperforms PACE here. In our modified scheduling mechanism, it only takes one RDS polling message to trigger a transmission burst. Therefore, the overhead of the polling messages going up and downwards through the multiple hops is avoided, which enables a better performance using FM-WiFIX, as can be seen in Fig. 5.14a. The differences in the throughout achieved by the various nodes is not a loss in the fairness but a positive discrimination of the nodes closer to the gateway. Note that the lowest throughput of our network is approximately equal to the average throughput achieved by PACE. This comes from the Eq. (4.2) that allows the nodes closer to the gateway to transmit more packets in their bursts. Note also that the nodes at one hop of the gateway could achieve higher throughputs. Due to the testbed limitations, if we generated too much traffic, some instances of the Iperf servers would start to fail so we had to limit the traffic generated. Therefore, we could not fully characterize the real throughput gains of our solution, in comparison with PACE, in these scenarios where the vast majority of traffic is generated at the sensor nodes and sent towards the gateway.



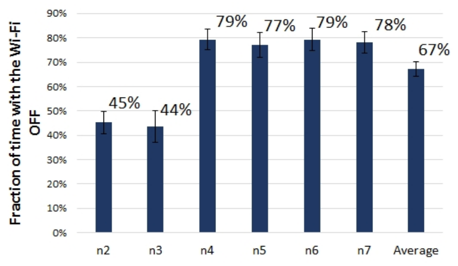
	n2	n3	n4	n5	n6	n7	Average
FM-WiFIX	356	352	340	341	338	340	345
PACE	314	315	314	314	314	314	314

(a) Comparison between the throughput achieved by PACE and FM-WiFIX in the binary tree topology.

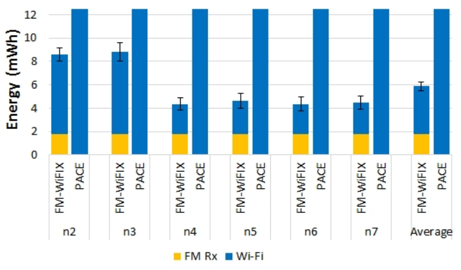


	n2	n3	n4	n5	n6	n7	Average
FM-WiFIX	71.47	33.95	80.36	81.86	62.81	43.24	62.28
PACE	5.10	5.06	4.11	4.89	5.13	4.46	4.79

(b) Comparison between the jitter presented by PACE and FM-WiFIX, in the binary tree topology.



(c) The amount of time that each node was with its Wi-Fi off, during the FM-WiFIX operation in the binary tree scenario.



(d) The average energy consumption of each node, during the 60 s of each experiment, with and without the FM-WiFIX optimization.

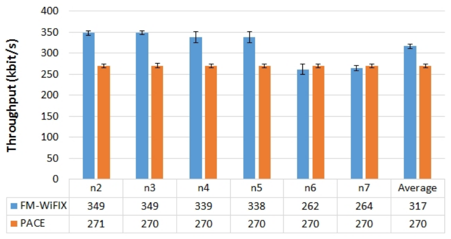
Figure 5.13: Results per node, for the **binary tree** topology (Fig. 5.12a).

As expected, this topology does not allow values of energy savings as high as the previous one (Fig. 5.14c). The jitter metric shows better results but still high variability. This metric will always be the worse in our solution due to the limitations of the RDS control channel.

5.5.2.3 Unbalanced Tree Topology

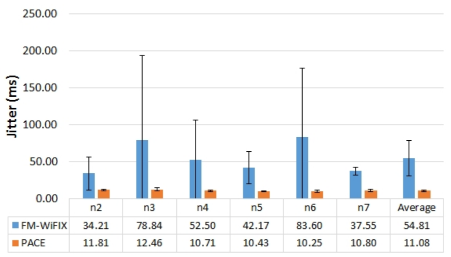
Regarding the system's throughput, within this scenario PACE and FM-WiFIX present almost similar results (Fig. 5.15a). This is due to the existence of three nodes at one hop and three nodes at two hops in this topology (see Fig. 5.12c). The modified scheduling mechanism leverages the network's throughput, relatively to PACE, with the increase of the number of hops to the gateway. Here, the total number of hops to the gateway is the minimum of the three scenarios, thus is normal that this is the scenario where PACE becomes closer to FM-WiFIX, concerning the throughput results.

The energy gains of this scenario are placed between the binary tree topology and the balanced tree with two branches and three hops. However, theoretically, this should be the scenario with the highest energy gains (Table 5.2). Maybe this network topology is less resilient to RDS errors or, during these tests, the RDS control channel was being affected by higher noise levels than in the binary tree tests.



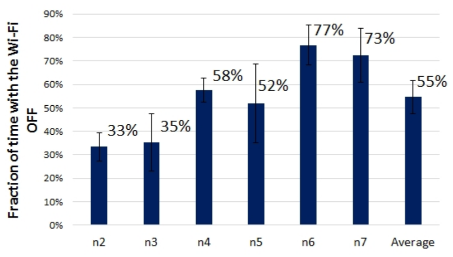
	n2	n3	n4	n5	n6	n7	Average
FM-WiFIX	349	349	339	338	262	264	317
PACE	271	270	270	270	270	270	270

(a) Comparison between the throughput achieved by PACE and FM-WiFIX in the balanced tree topology with 2 branches and 3 hops.

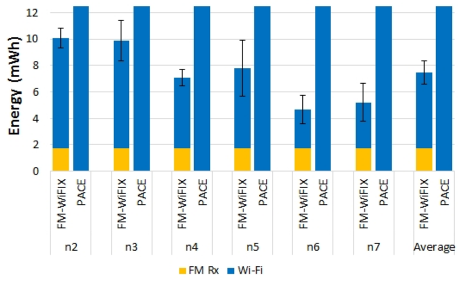


	n2	n3	n4	n5	n6	n7	Average
FM-WiFIX	34.21	78.84	52.50	42.17	83.60	37.55	54.81
PACE	11.81	12.46	10.71	10.43	10.25	10.80	11.08

(b) Comparison between the jitter presented by PACE and FM-WiFIX, in the balanced tree topology with 2 branches and 3 hops.

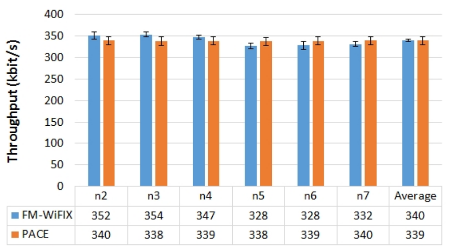


(c) The amount of time that each node was with its Wi-Fi off, during the FM-WiFIX operation in the balanced tree topology with 2 branches and 3 hops.



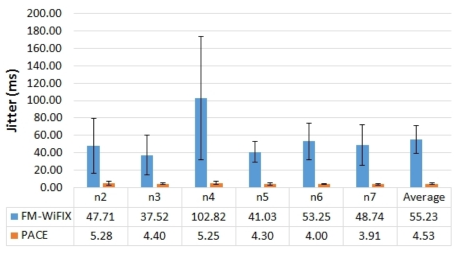
(d) The average energy consumption of each node, during the 60 s of each experiment, with and without the FM-WiFIX optimization.

Figure 5.14: Results per node, for the **balanced tree topology with two branches and three hops** (Fig. 5.12b).



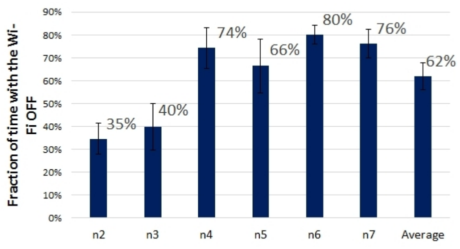
	n2	n3	n4	n5	n6	n7	Average
FM-WiFIX	352	354	347	328	328	332	340
PACE	340	338	339	338	339	340	339

(a) Comparison between the throughput achieved by PACE and FM-WiFIX in the unbalanced tree topology.

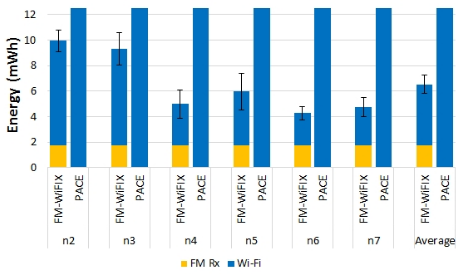


	n2	n3	n4	n5	n6	n7	Average
FM-WiFIX	47.71	37.52	102.82	41.03	53.25	48.74	55.23
PACE	5.28	4.40	5.25	4.30	4.00	3.91	4.53

(b) Comparison between the jitter presented by PACE and FM-WiFIX, in the unbalanced tree topology.



(c) The amount of time that each node was with its Wi-Fi off, during the FM-WiFIX operation in the unbalanced tree topology.



(d) The average energy consumption of each node, during the 60 s of each experiment, with and without the FM-WiFIX optimization.

Figure 5.15: Results per node, for the **unbalanced tree topology** (Fig. 5.12c).

5.5.3 Discussion

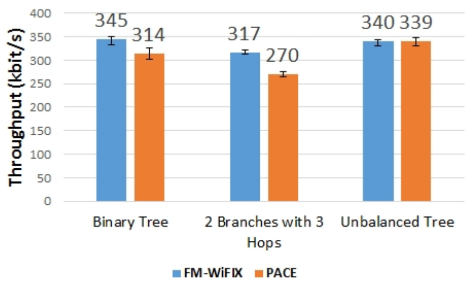
We now proceed to the discussion of the results obtained, starting with the presentation of some graphics that show the system's performance in the three scenarios tested.

In Fig. 5.16, we compare the throughput and jitter values obtained for both solutions, in each scenario. Our solution presents always a slightly better throughput than PACE. The new scheduling mechanism proposed, which consists on the assignment of time slots for transmission bursts, proposed in Section 4.4.2, is the responsible for these throughput gains. This is done at an apparent loss in fairness but we do not consider this new behavior as a loss. What happens is the following. In PACE, the fairness is based on giving the same number of transmission opportunities to every node in the network. Within this scheme, nodes closer to the gateway are limited by the increased transmission time of the nodes farther to it. There is fairness in throughput but unfairness in the transmission times assigned to each node.

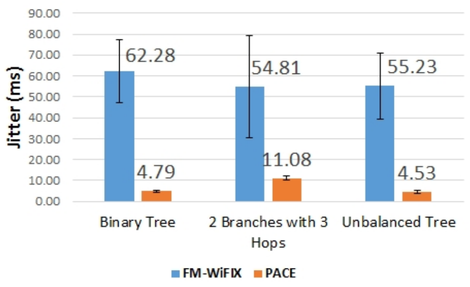
On the other hand, with the implemented transmission burst solution, now the **fairness consists on assigning to each node time slots of equal duration** for their transmission bursts. This implies that, during its time slot, the throughput of each node only depends of its distance to the gateway and is not influenced by the rest of the network, contrary to PACE. The network's topology only influences the interval between the transmission bursts of the same node, in function of the network's size. With this new scheme, every node is allowed to transmit at the maximum of its capacity, during its time slot, which allows better values for the system's average throughput. An additional contribution for the improvement of the system's throughput is the suppression of in-band control messages. Please note that this solution does not affect the throughput of the nodes farther to the gateway, it only removes the limitations imposed in PACE to the nodes that could achieve better throughputs. Note also that this solution was design specially for our video surveillance scenario, where the majority of traffic exchanged in the network is generated in the video sensors and sent towards the gateway. PACE was designed to different scenarios, with symmetrical traffic patterns so it is natural that our adjusted mechanism is able to outperform PACE in a WVSN.

Fig. 5.14a shows clearly the described differences between FM-WiFIX with the burst mechanism and PACE. With PACE all nodes present the same throughput of 270 kbit/s. With FM-WiFIX, the nodes at 3 hops from the gateway present ≈263 kbit/s, which is almost equal to PACE's average and the other nodes transmit at higher rates. The nodes at 2 hops present values of ≈338 kbit/s and the nodes at 1 hop reach the limit of ≈350 kbit/s, limit imposed to Iperf to avoid server failures due to the lack of processing power at the gateway, as explained in Section 5.5.1. Without this limitation, the throughput gains of our new solution would be more clearly observed in the other tested scenarios and would be significantly higher than the ones showed in this thesis. As future work, it would be interesting to properly quantify these gains, using a more suitable testbed.

The greatest weakness of our solution is undoubtedly the jitter values exhibited and its high variability, which is bad for a video stream scenario. The limitations of the RDS control channel are the responsible for this bad results. However, these jitter values would still enable a video stream transmission as, within video streaming applications, there are no significant jitter requirements because of application buffering [67].



(a) Comparison of the throughput values achieved by each solution in each scenario.



(b) Comparison between the jitter exhibited by PACE and FM-WiFIX in each topology.

Figure 5.16: An overview of the performance of the two solutions in the three scenarios considered Fig. 5.12.

Fig. 5.17 shows the average energy consumptions, associated to Wi-Fi, in the whole system, during the experiments of 60 s. We conclude that the best scenario is the binary tree topology, which was already expected. It is noteworthy that the average energy gains, achieved in the binary tree scenario, corroborate the simulation results, presented in [2]. Based on the first two bars of Fig. 5.17, we are able to calculate the system's average energy gain, associated to Wi-Fi, which is equal to $1 - \frac{39.36\,\mathrm{mWh}}{75\,\mathrm{mWh}} \approx 48\,\%$. Using the energy consumptions calculated for our system (Table 5.1) in the same simulation performed in [2], we got an estimation of 55 % for the energy gains in this binary tree topology, with 7 nodes. The small difference is caused by errors in the RDS control channel. However, the proximity between the simulation results and the practical results validates both our implementation and the original FM-WiFIX concept. Moreover, as the simulations results demonstrate an increase in the energy savings with the increase in the number of nodes, is also reasonable to expect that our results would be even better if we had assembled a bigger network. Note also that, in the worst scenario tested, we were still able to reduce in 33 % the energy associated with Wi-Fi consumption ($1 - \frac{50.36\,\mathrm{mWh}}{75\,\mathrm{mWh}} \approx 33\,\%$).

In Table 5.2, we present a comparison between the achieved and theoretical energy gains. The larger gap between these values in the second and third scenarios may indicate that or these topologies are more susceptible to control errors or higher noise levels were being introduced in the RDS channel during these tests. Nevertheless, these results show that our implementation is indeed capable of achieving decent energy savings and, by using a more reliable control channel, we could achieve even higher gains.

Concerning the power consumptions, in Fig. 5.18 we demonstrate the impact of our solution in the power consumption of our video sensors, based on the Raspberry Pi and also in the power consumption of dedicated video sensors (ASIC-based), where our improvements have a much larger expression. The latter was based on the results presented in [6], where the authors have isolated the power consumption of a specialized video sensor, running different video encoding

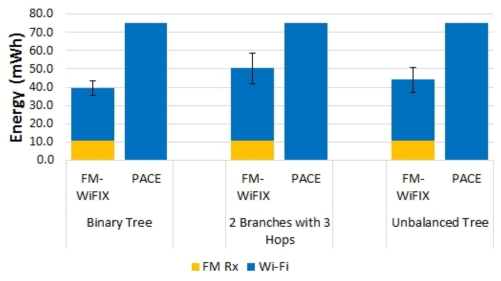
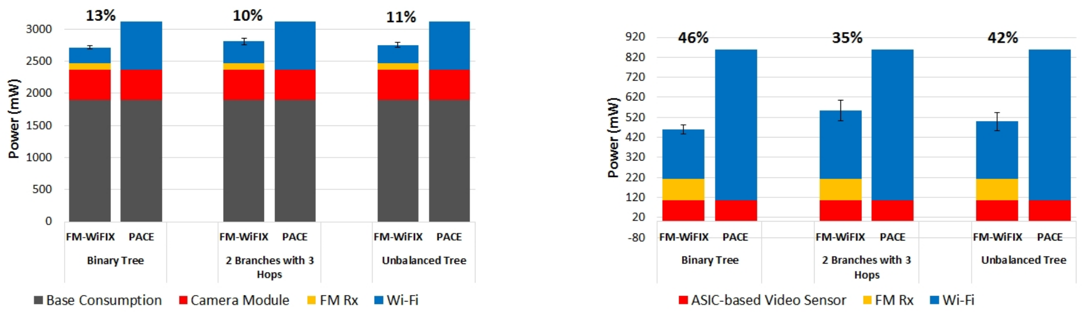


Figure 5.17: The average energy consumption, associated to Wi-Fi, in the whole system, during the 60 s of each experiment, with and without the FM-WiFIX optimization, for each scenario.

Scenario	Energy Gains	
	Achieved	Theoretical
Binary Tree	48%	58%
2 Branches with 3 Hops	33%	53%
Unbalanced Tree	41%	61%

Table 5.2: Energy gains for each scenario and the correspondent theoretical values.

algorithms, as showed in Fig. 2.5. The percentage of energy saved is also represented above the bars. Note that, in the second graph, using a more reliable control channel, we could easily achieve energy savings over 50 %, based on the theoretical values presented in Table 5.2.



(a) The impact of our solution in the overall power consumption of a video sensor based on the Raspberry Pi.

(b) The impact of our solution in the overall power consumption of an ASIC-based video sensor [6].

Figure 5.18: Impact of our solution in two different kinds of video sensors.

The evaluation of the FM-WiFIX implementation revealed some interesting results, specially regarding the energy savings and the throughput achieved, in comparison with PACE. Our proof-of-concept was effectively able to prove the validity of this proposal as it presents energy gains of up to 48 % in a typical topology while not only maintaining but also improving the performance enabled by PACE. Our new scheduling mechanism, proposed in Section 4.4.2, was the responsible for allowing such good results, regarding the system's throughput. This mechanism was designed to compensate the limitations of the RDS technology, and, as we conclude, although the final result using FM radio was satisfactory, to further improve the system, a more suitable technology for the requirements of the control channel should be employed.

Regarding this need, we have briefly studied some options, being one them the nRF24L01 ultra low power 2.4 GHz RF Transceiver IC [68], which consumes less than 14 mA ($\approx$ 47 mW) and seems to provide a good coverage area and large bandwidths. Moreover, this technology is able to form mesh networks, thus assuring the full coverage of our WVSN. For us, this seems to be the most promising alternative to the FM-RDS based control channel. Another option that we considered was using Bluetooth Low Energy (BLE), which meets all the requirements except the coverage range. This could be compensated with a flooding mechanism but this would bring extra challenges. An interesting feature of the BLE technology is the small size of the packets transmitted, which would be suitable for the control protocol designed [69]. Alternatively, following the initial idea of using Software Defined Radio (SDR) to completely change the RDS standard, we could be able to overcome the identified limitations. We could for example adjust the frame length to our control messages of 9 bit. However, this requires a full understanding of the Radio Data System and would be the same as specifying a new version of the Radio Data System, which is, understandably, out of the scope of this thesis.

We have also identified many practical problems that must be overcome. The testbed setup chosen is unacceptably limiting the throughputs available in our system, completely thwarting our initial goal of reproducing a real surveillance system. Nevertheless, the comparison with PACE leads to the conclusion that, with suitable hardware, our network would be able to support the transmission of multiple video streams with a good quality, as initially desired.

To increase the RDS polling rate, we have also compromised the initial range of the RDS channel, which reinforces once again the idea of choosing a different and more suitable technology for the control channel. In a real scenario, with the nodes physically apart from each other, our system would not be functional, as supported by the results presented in Section 5.4. However, the FM control channel implemented was useful for this proof of concept and could be used in another context, with fewer requirements regarding the variation frequency of the control messages or the control channel range.

Finally, the effective control over the state of the Wi-Fi card should also be accomplished, in order to fully implement the FM-WiFIX concept and achieve real gains. As explained in Section 5.3.1, accomplish this goal is not straightforward and this problematic alone could itself be the theme of another thesis.

Chapter 6

Conclusion

The purpose of this thesis was to implement the FM-WiFIX proposal on a real testbed, in order to perform a proof-of-concept of this solution. With this implementation, it was expected to address the three main problems of the Wireless Video Sensor Networks (WVSNs): throughput unfairness, bad performance and energy inefficiency.

The key concept on which this solution relies is the low power out-of-band control channel, RDS based. However, this solution is a rather ambitious one, that proposes the utilization of this out-of-band control channel in a way that, to the best of our knowledge, was never used before: signaling the transitions between on and off states of the Wi-Fi card in a matter of milliseconds.

Throughout our work, we have faced several challenges: the original FM-WiFIX had some problems that had to be solved to enable a successful implementation; the RDS technology was not the best choice to implement the control channel of the FM-WiFIX solution, which must be reliable, fast and provide a great coverage; the absence of support to effectively turn on and off the Wi-Fi radios; and the lack of processing power of the Raspberry Pi platform, that limited our network's performance.

Nevertheless, recurring to some turn around solutions and tweaks, we were able to successfully deploy a proof of concept prototype of a WVSN, implementing the FM-WiFIX proposal.

6.1 Contributions

With this thesis we have achieved many important contributions. We suggested and implemented multiple modifications to improve the FM-WiFIX concept, regardless of the technology used for the control channel (the control protocol, the register phase, the polling order and the rest of the modifications described in Section 4.2); we have characterized and adapt the Radio Data System to serve the purposes of the proposed control channel, although we have also noticed that the FM-RDS is not the best solution for the FM-WiFIX paradigm; we have designed a new scheduling mechanism for FM-WiFIX (the transmission burst scheme) that enables its operation with the limitations imposed by the RDS channel and is more suitable than PACE for the traffic pattern of a WVSN; we have characterized the power consumptions of the Raspberry Pi and the associated hardware; conducted a study about a multitude of paths to quickly turn on and off the Wi-Fi NIC; and finally, we concluded this work with an evaluation of the FM-WiFIX solution in several contexts, that effectively was this system's proof-of-concept as this evaluation has validated the simulation results obtained in [2].

6.2 Future Work

The main improvements that should be performed, in order to unlock the full potential of the FM-WiFIX solution and to achieve an optimized WVSN, for real time video streaming, are the following:

- It is mandatory to choose a new technology for the control channel. RDS is too slow, too unreliable and only achieves acceptable polling rates if we compromise its range. The Nordic's nRF24L01 [68] may be a good solution.
- From vital importance is also the identification and assembly of a new testbed, without using the Raspberry Pi model B. As previously described, this platform had become the bottleneck of our solution. With a suitable testbed, it would be possible to characterize the real throughput gains of this solution, in comparison with PACE.
- Find a solution that allows the effective transition between on and off states of the wireless NIC, without compromising the association to the network.
- As optimizations, we considered some options as frame aggregation throughout each branch, using jumbo frames, and spatial reuse to improve the system's throughput. Although we choose to spend our time studying other aspects of the solution, these are indeed very interesting paths of research to follow, in order to improve even more the FM-WiFIX solution.

References

[1] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[2] Filipe Sousa, Rui Campos, and Manuel Ricardo. Energy-efficient wireless multimedia sensor networks using FM as a control channel. In *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pages 1–7, June 2014. doi:10.1109/ISCC.2014.6912573.

[3] Ian F Akyildiz, Tommaso Melodia, and Kaushik R Chowdhury. A survey on wireless multimedia sensor networks. *Computer networks*, 51(4):921–960, 2007.

[4] Samina Ehsan and Bechir Hamdaoui. A survey on energy-efficient routing techniques with QoS assurances for wireless multimedia sensor networks. *Communications Surveys Tutorials, IEEE*, 14(2):265–278, Second 2012. doi:10.1109/SURV.2011.020211.00058.

[5] Vladimir Vukadinovic, Ioannis Glaropoulos, and Stefan Mangold. Enhanced power saving mode for low-latency communication in multi-hop 802.11 networks. *Ad Hoc Networks*, 23(0):18 – 33, 2014. URL: http://www.sciencedirect.com/science/article/pii/S1570870514001127, doi:http://dx.doi.org/10.1016/j.adhoc.2014.06.001.

[6] Shao-Yi Chien, Teng-Yuan Cheng, Shun-Hsing Ou, Chieh-Chuan Chiu, Chia-Han Lee, V Srinivasa Somayazulu, and Yen-Kuang Chen. Power consumption analysis for distributed video sensors in machine-to-machine networks. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 3(1):55–64, 2013.

[7] Using wireless mesh networks for video surveillance. White paper, Aruba Networks. URL: http://www.arubanetworks.com/pdf/technology/whitepapers/WP_VideoSurveillance.pdf.

[8] SRI Consulting Business Intelligence. Disruptive Civil Technologies: Six Technologies With Potential Impacts on US Interests Out to 2025. Technical report, National Intelligence Council, April 2008. Available: https://www.fas.org/irp/nic/disruptive.pdf [Accessed: Feb. 14, 2015].

[9] Deepanshu Shukla, Leena Chandran-Wadia, and Sridhar Iyer. Mitigating the exposed node problem in ieee 802.11 ad hoc networks. In *Computer Communications and Networks, 2003. ICCCN 2003. Proceedings. The 12th International Conference on*, pages 157–162. IEEE, 2003.

[10] Aruna Jayasuriya, Sylvie Perreau, Arek Dadej, and Steven Gordon. *Hidden vs exposed terminal problem in ad hoc networks*. PhD thesis, ATNAC 2004, 2004.

[11] Filipe Ribeiro, Rui Campos, David Rua, Carlos Pinho, and Jose Ruela. PACE: Simple multi-hop scheduling for single-radio 802.11-based stub wireless mesh networks. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on*, pages 103–110, Oct 2013. doi:10.1109/WiMOB.2013.6673347.

[12] Rui Campos, Ricardo Duarte, Filipe Sousa, Manuel Ricardo, and José Ruela. Network infrastructure extension using 802.1D-based wireless mesh networks. *Wireless Communications and Mobile Computing*, 11(1):67–89, 2011. URL: http://dx.doi.org/10.1002/wcm.916, doi:10.1002/wcm.916.

[13] Clicking Clean: How Companies are Creating the Green Internet. Technical report, Greenpeace, April 2014. Available: http://www.greenpeace.org/usa/Global/usa/planet3/PDFs/clickingclean.pdf [Accessed: Feb. 14, 2015].

[14] Daniel Halperin, Ben Greenstein, Anmol Sheth, and David Wetherall. Demystifying 802.11n power consumption. In *Proceedings of the 2010 international conference on Power aware computing and systems*, page 1. USENIX Association, 2010.

[15] Julio Leao da Silva Jr, Jason Shamberger, M Josie Ammer, Chunlong Guo, Suetfei Li, Rahul Shah, Tim Tuan, Michael Sheets, Jan M Rabaey, Borivoje Nikolic, et al. Design methodology for picoradio networks. In *Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001. Proceedings*, pages 314–323. IEEE, 2001.

[16] Eugene Shih, Paramvir Bahl, and Michael J Sinclair. Wake on wireless: an event driven energy saving strategy for battery operated devices. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 160–171. ACM, 2002.

[17] Lin Gu and John A Stankovic. Radio-triggered wake-up for wireless sensor networks. *Real-Time Systems*, 29(2-3):157–182, 2005.

[18] Ivaylo Haratcherev, Michele Fiorito, and Carine Balageas. Low-power sleep mode and out-of-band wake-up for indoor access points. In *GLOBECOM Workshops, 2009 IEEE*, pages 1–6. IEEE, 2009.

[19] Bas Van der Doorn, Winelis Kavelaars, and Koen Langendoen. A prototype low-cost wakeup radio for the 868 mhz band. *International Journal of Sensor Networks*, 5(1):22–32, 2009.

[20] Anthony Rowe, Rahul Mangharam, and Raj Rajkumar. Firefly: A time synchronized real-time sensor networking platform. *Wireless Ad Hoc Networking: Personal-Area, Local-Area, and the Sensory-Area Networks, CRC Press Book*, 2006.

[21] IEEE Std 802.11^{TM}-2012 (revision of IEEE Std 802.11-2007). *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 Am*, pages 1006–1009, March 2012.

[22] Adeel Shahzad, Ghalib A Shah, and Asif U Khattak. QoS-supported energy-efficient MAC (QEMAC) protocol based on IEEE 802.11 e for wireless multimedia sensor networks. In *Information Science and Service Science (NISS), 2011 5th International Conference on New Trends in*, volume 1, pages 200–204. IEEE, 2011.

[23] Quan Zhou, Yongjun Xu, and Xiaowei Li. HTSMAC: High throughput sensor MAC for wireless video networks. In *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pages 2428–2431. IEEE, 2007.

[24] Xinyu Zhang and Kang G Shin. E-mili: energy-minimizing idle listening in wireless networks. *Mobile Computing, IEEE Transactions on*, 11(9):1441–1454, 2012.

[25] Ivan Lee, William Shaw, and Jong Hyuk Park. On prolonging the lifetime for wireless video sensor networks. *Mobile Networks and Applications*, 15(4):575–588, 2010.

[26] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10 pp. vol.2–, Jan 2000. doi:10.1109/HICSS.2000.926982.

[27] Sibila Ratnaraj, Sarangapani Jagannathan, and Vittal Rao. OEDSR: Optimized energy-delay sub-network routing in wireless sensor network. In *Networking, Sensing and Control, 2006. ICNSC '06. Proceedings of the 2006 IEEE International Conference on*, pages 330–335, 2006. doi:10.1109/ICNSC.2006.1673167.

[28] Min Chen, Victor Leung, Shiwen Mao, and Yong Yuan. Directional geographical routing for real-time video communications in wireless sensor networks. *Computer Communications*, 30(17):3368–3383, 2007.

[29] Ilias Politis, Michail Tsagkaropoulos, Tasos Dagiuklas, and Stavros Kotsopoulos. Power efficient video multipath transmission over wireless multimedia sensor networks. *Mobile Networks and Applications*, 13(3-4):274–284, 2008.

[30] Praveen Gupta, Preeti Saxena, AK Ramani, and Rajkamal Mittal. Optimized use of battery power in wireless ad hoc networks. In *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*, volume 2, pages 1093–1097, Feb 2010.

[31] Cheong Seong Chee, A.B. Jambek, and R. Hussin. Review of energy efficient block-matching motion estimation algorithms for wireless video sensor networks. In *Computers Informatics (ISCI), 2012 IEEE Symposium on*, pages 241–246, March 2012. doi:10.1109/ISCI.2012.6222702.

[32] Geert Van der Auwera, Prasanth David, and Martin Reisslein. Traffic characteristics of H. 264/AVC variable bit rate video. *Communications Magazine, IEEE*, 46(11):164–174, 2008.

[33] Demostenes Zegarra Rodriguez and Graca Bressan. Performance assessment of high efficiency video coding-HEVC. In *Global High Tech Congress on Electronics (GHTCE), 2013 IEEE*, pages 110–111. IEEE, 2013.

[34] bridge | the linux foundation. [Online]. Available: http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge. [Accessed: Jun. 25, 2015].

[35] Ahmad Rahmati, Lin Zhong, Venu Vasudevan, Jehan Wickramasuriya, and Daniel Stewart. Enabling pervasive mobile applications with the FM radio broadcast data system. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, pages 78–83. ACM, 2010.

[36] NORME EUROPÉENNE. Specification of the radio data system (rds) for vhf/fm sound broadcasting in the frequency range from 87,5 to 108,0 mhz. April 1998. Available: http://www.interactive-radio-system.com/docs/EN50067_RDS_Standard.pdf [Accessed: Jun. 28, 2015].

[37] Christopher P Paolini, Christopher Aguirre, Mahasweta Sarkar, and Santosh Nagaraj. An ad hoc broadcast scheme using the radio data system (RDS) in the FM band. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, 2013.

[38] GNU general public license, version 3. [Online]. Available: http://www.gnu.org/licenses/gpl.html, June 2007. [Accessed: Feb. 9, 2015].

[39] Andrew Tanenbaum and Maarten Van Steen. *Distributed systems: principles and paradigms*. Pearson Prentice Hall, 2 edition, 2007.

[40] Christophe Jacquet. FM-RDS transmitter using the raspberry pi's PWM. [Online]. Available: https://github.com/ChristopheJacquet/PiFmRds. [Accessed: Feb. 9, 2015].

[41] Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman. *Linux device drivers*. O'Reilly Media, Inc., 3 edition, 2005.

[42] W Richard Stevens and Stephen A Rago. *Advanced programming in the UNIX environment*. Addison-Wesley, 3 edition, 2013.

[43] GNU Radio. [Online]. Available: http://gnuradio.org/redmine/projects/gnuradio. [Accessed: Feb. 10, 2015].

[44] SparkFun. *SparkFun FM Tuner Evaluation Board - Si4703. [Online]. Available: https://www.sparkfun.com/products/10663*. [Accessed: Jun. 18, 2015].

[45] RdSpi, a Si4703 based RDS scanner for the raspberry pi. [Online]. Available: https://github.com/achilikin/RdSpi. [Accessed: Jun. 18, 2015].

[46] Broadcom. *High Definition 1080p Embedded Multimedia Applications Processor BCM2835. [Online]. Available: http://www.broadcom.com/products/BCM2835*. [Accessed: Feb. 9, 2015].

[47] Premier Farnell UK Limited. *Raspberry Pi, Model B, 512MB SDRAM. [Online]. Available: http://cpc.farnell.com/1/1/95808-raspberry-pi-model-b-512mb-sdram-raspbrry-pcba-raspberry-pi.html*. [Accessed: Feb. 9, 2015].

[48] Simple guide to the RPi GPIO header and pins [Online]. Available: http://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/. [Accessed: Feb. 9, 2015].

[49] Raspbian. [Online]. Available: http://www.raspbian.org/. [Accessed: Feb. 10, 2015].

[50] TP-LINK. *300Mbps Mini Wireless N USB Adapter TL-WN823N. [Online]. Available: http://www.tplink.com/resources/document/TL-WN823N_V1_datasheet.pdf*. [Accessed: Feb. 9, 2015].

[51] 150Mbps high gain wireless USB adapter TL-WN722N. [Online]. Available: http://www.tp-link.com/en/products/details/cat-11_TL-WN722N.html. [Accessed: Jun. 22, 2015].

[52] Camera module for the raspberry pi. [Online]. Available: https://www.raspberrypi.org/products/camera-module/. [Accessed: Jun. 20, 2015].

[53] raspivid - raspberry pi documentation. [Online]. Available: https://www.raspberrypi.org/documentation/usage/camera/raspicam/raspivid.md. [Accessed: Jun. 20, 2015].

[54] Streaming video using VLC player. [Online]. Available: http://www.raspberry-projects.com/pi/pi-hardware/raspberry-pi-camera/streaming-video-using-vlc-player. [Accessed: Jun. 20, 2015].

[55] Iperf - The TCP/UDP bandwidth measurement tool. [Online]. Available: https://iperf.fr/. [Accessed: Feb. 11, 2015].

[56] C. Demichelis and P. Chimento. IP packet delay variation metric for IP performance metrics (IPPM). RFC 3393, IETF, November 2002. Available: http://tools.ietf.org/html/rfc3393 [Accessed: Feb. 11, 2015].

[57] YKUSH - yepkit USB switchable hub. [Online]. Available: https://www.yepkit.com/products/ykush. [Accessed: Jun. 21, 2015].

[58] mac80211 overview. [Online]. Available: http://linuxwireless.org/attachments/en/developers/Documentation/mac80211/mac80211.pdf. [Accessed: Jun. 21, 2015].

[59] mac80211 source code. [Online]. Available: https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/net/mac80211?id=refs/tags/v4.1-rc1. [Accessed: Jun. 21, 2015].

[60] Mesh power save implementation notes. [Online]. Available: https://github.com/o11s/open80211s/wiki/Mesh-Powersave-Implementation-Notes. [Accessed: Jun. 25, 2015].

[61] Kyle Jamieson. *Implementation of a power-saving protocol for ad hoc wireless networks*. PhD thesis, Massachusetts Institute of Technology, 2002.

[62] Wifi module - ESP8266. [Online]. Available: https://www.sparkfun.com/products/13252. [Accessed: Jun. 21, 2015].

[63] Atheros. *AR9271 Single-Chip 1x1 MAC / BB / Radio / PA / LNA with USB Interface for 802 . 11n 2 . 4 GHz WLANs. [Online]. Available: http://www.cqham.ru/forum/attachment.php?attachmentid=155133&d=1383397504*. [Accessed: Jun. 21, 2015].

[64] Tektronix. *Tektronix DMM4050 and DMM4040 Digital Multimeters Datasheet. [Online]. Available: http://www.tek.com/datasheet/dmm4050-4040-digital-multimeter*. [Accessed: Jun. 22, 2015].

[65] Usb made simple part 2 - electrical. [Online]. Available: http://www.usbmadesimple.co.uk/ums_2.htm. [Accessed: Jun. 22, 2015].

[66] Wifi analyzer - android apps on google play. [Online]. Available: https://play.google.com/store/apps/details?id=com.farproc.wifi.analyzer. [Accessed: Jun. 29, 2015].

[67] Tim Szigeti and Christina Hattingh. *End-to-end QoS network design: Quality of Service in LANs, WANs, and VPNs*. Cisco press, 2005.

[68] Nordic Semiconductor. *nRF24L01 - 2.4GHz RF - Products - Nordic Semiconductor. [Online]. Available: https://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01*. [Accessed: Jun. 28, 2015].

[69] Bluetooth low energy - technical information. [Online]. Available: http://www.bluetooth.com/Pages/low-energy-tech-info.aspx. [Accessed: Jun. 29, 2015].