Masters in Informatics Engineering
Master Thesis
Final Report

Towards Energy Efficient Multimedia Streaming in Mobile Devices

Bruno Filipe Ferreira Correia
bcorreia@student.dei.uc.pt

Advisors:
Marília Curado
Vítor Bernardo
Date: 1st of September, 2014

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Towards Energy Efficient Multimedia Streaming in Mobile Devices

Master Thesis submitted to the University of Coimbra

Author:
Bruno Filipe Ferreira Correia

Advisors:
Marília Pascoal Curado
Vítor Manuel H. Bernardo

September 1, 2014

I would like to dedicate this thesis to my parents.

Thank you for all the support.

Acknowledgements

First, I would like to express my gratitude to my adviser, Marília Curado, for supporting me all the time. Thank you for being always available and providing me with excellent suggestions and comments. Your confidence in my work was essential to motivate me and made it possible to improve my work constantly. Thank you for everything.

To Vítor Bernardo, my co-adviser, a big thanks for being always present, for introducing me to the LCT and for being constantly willing to clarify any of my doubts. Without your knowledge and dedication in this work, this thesis would not be possible.

Thank you to all my friends that directly or indirectly have always been present throughout my life, each one of you are responsible for making the person that I am.

A very special thanks to Carlos Ferreira for the friendship over all these years. The weeks that you were in Portugal were always the best ones for me. I hope that we can spend more time together soon.

I would like to thank Rafael Lopes for the friendship, for the conversations and for everything else that a friend could ask.

Thank you also João Monteiro for all the funny moments and for the good times without worries.

A big thanks to Mário Gago and Gil Hilário for being those friends that I will never forget. For being always there with your amazing sense of humor, for hosting me in Algarve and also in Coimbra. For all the crazy things and much more, "those were the best days of my life".

To Sara Monteiro, whose words cannot show how much I thank you for all these years, thanks for being always present, in the good and the bad moments, even when you are far away. For listening to me, for helping me, for supporting me and for believing in me. But, the most important, for being the person you are.

A special thanks also to Mariana Lourenço who became a big support in the last years. For all the good moments and for pushing me up when the things were going wrong. Thank you also for making me feel more confident by believing in me and supporting all of my decisions. Thank you for everything.

Last but not least, to my family, especially to my parents who have been providing me with the best conditions in life. Thank you for being the best parents a son could have.

Abstract

In the last few years the mobile devices have been increasingly used as part of the people daily routines. The Android smartphones and tablets are responsible for a large part of this increase and are used around the clock, causing the energy consumption in these devices to be a concern. Since these devices are most of the time connected to the Internet, the energy consumed by the IEEE 802.11 interface is responsible for a fast drain in the battery lifetime, leading to the need of finding solutions to reduce the wireless interface energy consumption.

Therefore, this work presents the state of the art concerning the IEEE 802.11 technology and a discussion of the most important energy consumption optimizations presented in the literature. To allow the study of the energy consumption in Android devices, a testbed setup is described and was used to perform a characterization of the IEEE 802.11 interface energy consumption in Android devices. The results showed that the implemented power saving techniques do not provide a proper trade-off between the energy consumption and the end-user expectations in the presence of Continuous Media Applications. Since the Continuous Media Applications are expected to be responsible for the most part of the mobile traffic in a few years, proposing mechanisms that keep the end-user expectations while reducing the energy consumption becomes a challenge.

With this challenge in mind, this work presents the EXPoSE framework which aims to extend the control of the IEEE 802.11 interface to the end-users, in Android devices. The validation of this framework in a real testbed showed energy savings between 5% and 54% while taking into account the end-user configurations.

Furthermore, this work also proposes the enhanced Power save Algorithm for continuous Media Applications (OPAMA) lite mechanism for Android devices. The experimental evaluation in a real testbed showed a clear benefit

of employing this mechanism in the presence of Continuous Media Applications, since, it is possible to reduce more than 55% of the energy consumption while keeping the end-user expectations, namely the maximum allowed delay.

Keywords: IEEE 802.11, Continuous Media Applications, Android, Energy, End-Users.

Contents

List of Figures

List of Tables

Abbreviations and acronyms

3G	Third Generation of mobile phone standards and technology
AC-Power	Alternative Current Power
Adaptive-PSM	Adaptive Power Save Mode
AP	Access Point
API	Application Programming Interface
AM	Active Mode
A-MSDU	Aggregate Medium Access Control (MAC) Service Data Unit
A-MPDU	Aggregate MAC Protocol Data Unit
ACK	Acknowledgment
APSD	Automatic Power-Saving Delivery
CAP	Controlled Access Phase
CF-Poll	Contention-Free Poll
CLBB	Cross-Layer Burst Buffering
CP	Contention Period
CPU	Central Proccessing Unit
CFP	Contention Free Period
CS	Carrier Sense
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTS	Clear to Send
DCF	Distributed Coordination Function
D-ITG	Distributed Internet Traffic Generator
DSSS	Direct Sequence Spread Spectrum

DTT	Downlink Transmission Time
EDCA	Enchanced Distributed Channel Access
ECDF	Empirical Cumulative Distribution Function
FHSS	Frequency Hopping Spread Spectrum
FIFO	First In, First Out
GHz	Gigahertz
HAL	Hardware Abstraction Layer
HCCA	Hybrid Coordination Function (HCF) Controlled Channel Access
HCF	Hybrid Coordination Function
IEEE	Institute of Electrical and Electronics Engineers
IR	Infrared
IPC	Inter Process Communication
JNI	Java Natural Interface
KLS	Kernel Level Shaper
Legacy-PSM	Legacy Power Save Mode
MAC	Medium Access Control
MLME	MAC sublayer Management Entity
Mbps	Megabit per second
MIMO	Multiple-Input and Multiple-Output
NAV	Network Allocation Vector
NDK	Native Development Kit
No-PSM	No Power Save Mode
ntpd	Network Time Protocol daemon
OPAMA	Power save Algorithm for continuous Media Applications
OS	Operating System
PCF	Point Coordination Function
PCH	Pagin Channel
PHY	Physical
PLCP	Physical Layer Convergence Procedure

PMD	Physical Medium Dependent
PS-Poll	Power-Save Poll
PSM	Power Save Mode
PSMP	Power Save Multi-Pool
QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
RLC	Radio Link Control
RTP	Real-Time Transport Protocol
RTT	Round Trip Time
RTS	Request to Send
RTS/CTS	Request to Send/Clear to Send
SAPSM	Smart Adaptive PSM
SCPI	Standard Commands for Programmable Instruments
SDK	Software Development Kit
S-APSD	Scheduled Automatic Power-Saving Delivery (APSD)
SP	Service Period
TCP	Transport Control Protocol
TIM	Traffic Indication Map
TXOP	transmission opportunities
U-APSD	Unscheduled APSD
UDP	User Datagram Protocol
UTT	Uplink Transmission Time
USB	Universal Serial Bus
VoIP	Voice over IP
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network

Chapter 1

Introduction

This work presents the development and validation, in Android devices, of two mechanisms based on OPAMA proposed by Bernardo et al. [Bernardo et al., 2013], namely the OPAMA lite and the enhanced OPAMA lite. Furthermore, a framework which gives full control of the IEEE 802.11 interface to the end-users and/or applications, in Android devices, is also proposed in this thesis.

This chapter is organized as follows: the first section presents the motivation for this work, followed by the identification of the main objectives to be achieved. The contributions are provided next and, finally, the thesis structure is presented.

1.1 Motivation

We live today in a technological era where almost every device has a wireless technology interface. In fact, there are multiple wireless technologies available nowadays. Nevertheless, the IEEE 802.11 technology is the most implemented and it is present in several portable devices.

With the growing number of mobile devices in the last years, anyone has access to the Internet everywhere through IEEE 802.11 interfaces, whether at home, in a coffee shop, a shopping area or even outdoors with the creation of hotspots. This allows the users to be always connected and, consequently, also leads to a lower battery lifetime in devices.

The Android platform is responsible for a large part of this growth, since 1 million Android devices are activated every day [Android, 2014a]. In fact, the tablets and smartphones with the Android Operating System (OS) are used in everyday life, however, the expectations by the end-user regarding the battery lifetime of these devices are not

always achieved. Since the device is with the user for 24 hours a day, it is expected that the battery lasts at least a full day, which does not always happen. This occurs in part, because the device is constantly connected to the internet, causing the IEEE 802.11 interface to be a bottleneck concerning energy consumption.

Moreover, it is expected that in 2017, the Continuous Media Applications represent the largest slice of the mobile data traffic, over 66% [Cisco, 2013]. Therefore, efforts should be made to create energy efficiency mechanisms in the presence of continuous traffic, allowing the end-users to achieve the best trade-off between energy consumption and performance, when using the IEEE 802.11 interface.

1.2 Objectives

This work aims to contribute to a green wireless communication by overcoming the existing gaps of the IEEE 802.11 technology in Android devices, namely the lack to provide the full control of the wireless interface and the inability to achieve a good trade-off between the energy consumption and the end-user expectations in the presence of Continuous Media Applications. Therefore, the goals of the work presented in this thesis, consist in developing a framework to allow the full control of the IEEE 802.11 interface by the end-users and the implementation of mechanisms which allow energy savings in the presence of continuous traffic while keeping the desired end-user expectations, in Android devices.

1.3 Contributions

During the development of this Master Thesis, the author performed some important contributions concerning the IEEE 802.11 technology in Android devices, as showed next.

This work presents a detailed analysis of the Institute of Electrical and Electronics Engineers (IEEE) 802.11 technology, with a description of the power saving techniques concerning the MAC layer and the current power management modes implemented in Android devices.

A study of the state of the art is also presented, where various mechanisms proposed in the literature to optimize the IEEE 802.11 interface are described and compared. Furthermore, the works concerning the assessment of the IEEE 802.11 interface energy consumption are also described and compared, with a focus in the methodologies developed for Android devices.

In order to evaluate the IEEE 802.11 interface in Android devices, this work describes a testbed based on two works presented in the literature: a methodology to measure the energy consumption of a network interface with high precision, proposed by Bernardo et al. [Bernardo et al., 2011], and a methodology to assess the energy consumption of the IEEE 802.11 interface in Android devices, proposed by Rice et al. [Rice and Hay, 2010].

Through the testbed above mentioned, the IEEE 802.11 interface in Android devices was fully tested. Therefore, this thesis presents a characterization of the current developed power management modes of the IEEE 802.11 technology in Android devices.

A framework to extend the default implementation of the IEEE 802.11 technology in Android devices is also presented. Moreover, in addition to providing the full control of the wireless interface by the end-users, this framework also provides two mechanisms to reduce the energy consumption of the Android devices by taking into account the end-user configurations.

The author developed and validated two mechanisms based on the Power save Algorithm for continuous Media Applications (OPAMA), proposed by Bernardo et al. [Bernardo et al., 2013], the OPAMA lite and the enhanced OPAMA lite. The OPAMA lite accomplishes a solution that extends the standard Power Save Mode (PSM) to allow the end-users to configure the maximum delay tolerated. The enhanced OPAMA lite was developed to improve the OPAMA lite performance by avoiding the polling phase and the conducted experimental evaluation showed that it is possible to reduce significantly the energy consumption in the presence of Continuous Media Applications, while keeping the end-user expectations most of the times.

In the course of this Master Thesis, an abstract concerning this work was submitted and accepted by the "*Rede Temática de Comunicações Móveis*" (RTCM) [RTCM, 2014]. This abstract can be found in Appendix A and leaded to a presentation in the *18º Seminário da RTCM*.

The work performed in this thesis also lead to a publication of a paper named *Towards End-User Driven Power Saving Control in Android devices* [Bernardo et al., 2014] that can be found in Appendix A. This paper was presented at *The 14th International Conference on Next Generation Wired/Wireless Advanced Networking* (NEW2AN 2014).

Furthermore, the author of this work plans to submit another paper to the *30th Annual ACM Symposium on Applied Computing SAC 2015 (ACM SAC 2015)*, concerning the experimental evaluation of the OPAMA mechanisms development for Android devices.

1.4 Thesis Structure

This section introduces the structure of this Master Thesis. Firstly, Chapter 2 presents the needed background concerning the IEEE 802.11 technology. In Chapter 3, various proposals regarding energy efficiency and energy assessment in IEEE 802.11, presented in the literature, are described and compared between each other. Chapter 4 presents an experimental evaluation concerning the IEEE 802.11 technology which allows the characterization of the IEEE 802.11 interface power management modes developed in Android devices. Moreover, the testbed setup is also described. The framework to extend the control of the IEEE 802.11 interface to the end-users in Android devices is described and validated in Chapter 5. The both implementations of the OPAMA mechanism for Android devices are discussed and evaluated in Chapter 6. Finally, Chapter 7 illustrates and describes the work plan and Chapter 8 presents the final conclusions.

Chapter 2

IEEE 802.11 Technology

This chapter presents the needed background regarding the IEEE 802.11 technology. Section 2.1 describes the IEEE 802.11 standard, including the most important versions and the most significant improvements. The main layers are described next, with a brief explanation about the Physical (PHY) layer, in Section 2.2, and a description about the MAC layer, in Section 2.3. Finally, Section 2.4 describes the power management modes in IEEE 802.11.

2.1 IEEE 802.11 Standard

The IEEE 802.11 standard defines the Wireless Local Area Network (WLAN) by specifying one Medium Access Control (MAC) layer and several Physical (PHY) layers. The IEEE LAN/MAN Standards Committee is responsible for the development and maintenance of the networking standards for local and metropolitan area networks. Whereas, the IEEE 802.11 Working Group is the entity responsible for the IEEE 802.11 standard.

The original version of the IEEE 802.11 standard [IEEE, 1997] dates from 1997. Actually, this version is known as legacy IEEE 802.11, since it specifies only bit rates of 1 or 2 Megabit per second (Mbps). In 1999 the first revision of the IEEE 802.11 standard was published with the IEEE 802.11a [IEEE, 1999a] and IEEE 802.11b [IEEE, 1999b] amends. The IEEE 802.11b uses the same band as legacy IEEE 802.11 (2.4 Gigahertz (GHz)), however it supports bandwidth up to 11 Mbps. On the other hand, the IEEE 802.11a uses the 5.0 GHz band and supports bandwidth up to 54 Mbps.

Another major revision was published in 2007 [IEEE, 2007] which incorporates various amends. The most important were IEEE 802.11g [IEEE, 2003], from 2003, and IEEE 802.11e [IEEE, 2005], from 2005. The IEEE 802.11g aims to combine the best of

IEEE 802.11a and IEEE 802.11b, specifying a standard that uses the 2.4 GHz band and supports bandwidth up to 54 Mbps. The IEEE 802.11e amend, defines enhancements regarding the MAC layer and introduces Quality of Service (QoS) capabilities.

The current revision was published in 2012 [IEEE, 2012] and, as the 2007 revision, it incorporates various amends. The most relevant is the IEEE 802.11n [IEEE, 2009], from 2009, that contains enhancements for higher throughput, namely by introducing the support for Multiple-Input and Multiple-Output (MIMO) (i.e., both transmitter and receiver can use multiple antennas to communicate). Additionally, it also defines the Power Save Multi-Pool (PSMP) mechanism and two new aggregation approaches, the Aggregate MAC Service Data Unit (A-MSDU) and the Aggregate MAC Protocol Data Unit (A-MPDU).

2.2 Physical Layer (PHY)

The IEEE 802.11 standard [IEEE, 2012] defines a set of Physical layers, in order to provide wireless connectivity in a local area network for fixed stations, portable and in movement. Since the PHY layer is beyond the scope of this work, only a brief introduction will be done.

The PHY layer has two main components: the Physical Layer Convergence Procedure (PLCP) and the Physical Medium Dependent (PMD). The PLCP uses media access techniques for mapping the transmitted/received data units into a framing format, while the PMD defines the characteristics and methods of the transmitted/received data.

In order to allow the data exchange, the IEEE 802.11 standard also defines some techniques such as the Infrared (IR), Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS).

2.3 Medium Access Control (MAC) Layer

The MAC layer specified in the IEEE 802.11 standard [IEEE, 2012] is a sub-layer of the Data Link layer. It contains a set of techniques which aim to manage the use of a shared medium. The MAC layer is responsible for three main features: reliable data delivery, access control and security. Additionally, the MAC layer also grants the communication with the upper layers.

Figure 2.1 illustrates the MAC layer architecture. The main existing components are explained in the following subsections.

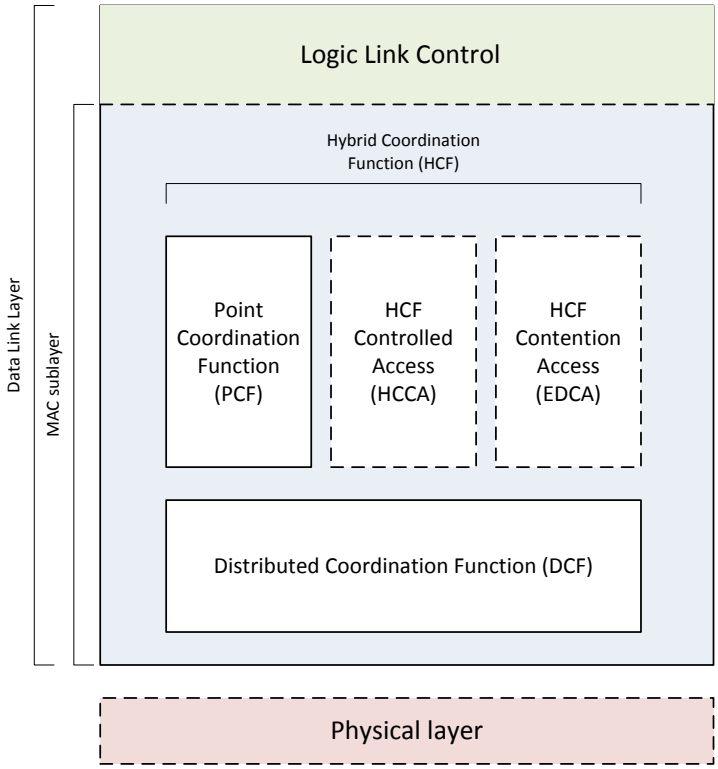


Figure 2.1: MAC layer architecture. Based on [IEEE, 2012]

2.3.1 Distributed Coordination Function (DCF)

The Distributed Coordination Function (DCF) [IEEE, 2012] is an access function which allows the automatic medium sharing between physical layers. The DCF employs the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) and a random backoff time.

Since the access to the carrier is based on contention, the CSMA/CA mechanism was designed to reduce the probability of collisions. When a station wants to transmit, it needs firstly to check if no other station is transmitting (i.e., verify if the channel is *idle*). The Carrier Sense (CS) is the mechanism responsible to verify if a channel is *idle* or *busy* and needs to be performed physically and virtually. The physical verification is addressed by the PHY layer that senses the channel to see if there is any activity. On the other hand, the MAC layer is responsible for the virtual verification. The virtual verification is done through the *Network Allocation Vector (NAV)* that indicates for how long the frame will use the channel. If the *NAV* informs that the channel is *idle* and the physically verification the same, the CS returns *idle*, otherwise, it returns *busy*.

When the channel stays *idle* after a *busy* period, the probability of collisions is higher, because many stations could be waiting for the medium to be available. To avoid such problem, stations cannot transmit immediately, and must use a random backoff time that will be decremented while the medium is *idle*. When this counter reaches zero the station is able to transmit. To transmit continuous frame sequences, the station needs to verify if the channel will stay *idle* for the whole transmitting time. Otherwise, the station needs to wait for the end of the current transmission.

Nevertheless, collisions may occur in the case of a hidden node. The hidden node problem occurs when one station (B) is between two stations (A and C) and the stations A and C cannot hear each other. Therefore, the stations A and C can send frames to station B at the same time and a collision occurs. In order to solve this problem, the Request to Send/Clear to Send (RTS/CTS) mechanism can be used. This mechanism allows the station to send a Request to Send (RTS) before transmit. The station that receives the RTS can hear the other stations and may send a Clear to Send (CTS) as a response to allow the station to transmit.

The DCF is also the base of other MAC layer mechanisms such as Point Coordination Function (PCF) and Hybrid Coordination Function (HCF), as described in the following subsections.

2.3.2 Point Coordination Function (PCF)

The Point Coordination Function [IEEE, 2012] is a method that allows the AP to control the access to the medium by introducing Contention Free Periods (CFPs).

In PCF the AP acts as a coordinator and defines which station is able to transmit by polling the stations. The station that receives a *Power-Save Poll (PS-Poll)* frame is able to exchange packets with the AP. If the station has a packet to transmit, it just

needs to transmit the packet. Otherwise, the station needs to inform the AP that it does not have packets to transmit.

Since the Access Point can use the PCF together with the DCF, it needs to initiate Contention Periods (CPs) and Contention Free Periods (CFPs) over the time. In the CP the Distributed Coordination Function is employed. On the other hand, the CFP uses the Point Coordination Function. If a station wants to be polled by the AP, in order to use the CFPs, it needs to send a Contention-Free Poll (CF-Poll) frame to the AP.

The PCF allows the stations to avoid the contentions present in DCF and also to avoid mechanisms such as RTS/CTS that introduce additional overhead. However, the PCF is an optional technique since its implementation is not mandatory. In fact, only a restricted group of commercial APs and WLAN interface cards implement it [Tsao and Huang, 2011].

2.3.3 Hybrid coordination function (HCF)

The Hybrid Coordination Function (HCF) is an additional coordination function that introduces QoS. The HCF uses the Distributed Coordination Function (DCF) and Point Coordination Function (PCF) with QoS mechanisms and new frame sub-types. This allows the HCF to enable QoS data transfers during Contention Periods and Contention Free Periods.

In HCF the Enchanced Distributed Channel Access (EDCA) is used for a contention-based channel access during a Contention Period and the HCF Controlled Channel Access (HCCA) mechanism is used for a controlled channel access during a Contention Free Period. Both EDCA and HCCA mechanisms were defined in IEEE 802.11e [IEEE, 2005].

The EDCA mechanism defines four DCF variants, which represent classes for background traffic, best effort traffic, video and voice. Each class contains a different set of parameters depending on the QoS needs of a specific traffic type. This allows the high priority traffic to have transmission opportunities (TXOP) before the low priority traffic.

The HCCA mechanism uses a coordinator that is able to allocate Contention Free Periods, named by Controlled Access Phase (CAP), in a Contention Period. In CAP the Access Point, which is usually the coordinator, manages the radio resources to ensure transmission opportunities to the stations that want to access the channel.

2.4 IEEE 802.11 Power Management Modes

Due to the need of many stations which use the IEEE 802.11 technology to be energy efficient, the IEEE 802.11 standard [IEEE, 2012] was designed to meet power saving requirements.

The standard defines that a station can operate in two modes concerning energy saving, the Active Mode and the Power Save Mode. In the Active Mode (AM), also called by No-PSM, the station is fully powered by setting the wireless interface to be continuously awake, i.e. the station radio is always on, listening to the channel. This allows a station to transmit or receive packets at any time, ensuring high availability and performance. However, this mode is responsible for a high energy consumption. In Power Save Mode (PSM) a station is able to sleep for certain periods. In this sleep period, known by the name of doze state, the station can turn off most of the hardware radio components consuming very low energy. However, when in this state, it does not listen to the channel.

2.4.1 Power Save Mode (Legacy-PSM)

According to the IEEE 802.11 standard [IEEE, 2012] an AP defines a *Beacon* interval (usually of 100 ms) and periodically broadcasts *Beacon* frames. A station operating in PSM should wake up to receive the *Beacon* frames. If the station does not have packets to transmit or receive it can go to sleep.

When a station wants to operate in PSM, it firstly needs to inform the AP by sending a *NULL Data* frame. The frame contains a *Power Management* bit (in the *Frame Control* field) that is set to 1 if the station wants to use the PSM. If the bit is set to 0, the station wants to use the No-PSM mode to stay continuously awake. After the *NULL Data* frame is sent by the station, the AP replies with an *Acknowledgment (ACK)* and the station can change its mode.

The AP is responsible for buffering all the downlink packets to the stations that are sleeping. A station operating in PSM wakes up to receive the *Beacon* frames which contain a Traffic Indication Map (TIM). The TIM informs if the AP has buffered packets to the station. If the AP has buffered packets, the station needs to contend for the channel and send a *PS-Poll* frame, in order to receive the pending data. These downlink packets are sent to the station individually, right after the *PS-Poll* frame. Otherwise, the AP sends an *ACK* to the station and delivers the packet later. The packets contain a *MORE* field that indicates if there are more packets buffered in the AP. If the *MORE* field is present, the station continues to send *PS-Polls* frames for every buffered packet

until all the packets are delivered.

Figure 2.2 illustrates the communication between an AP and a station when the station changes its mode to operate in Legacy-PSM. Moreover, the energy consumption by the station is also depicted.

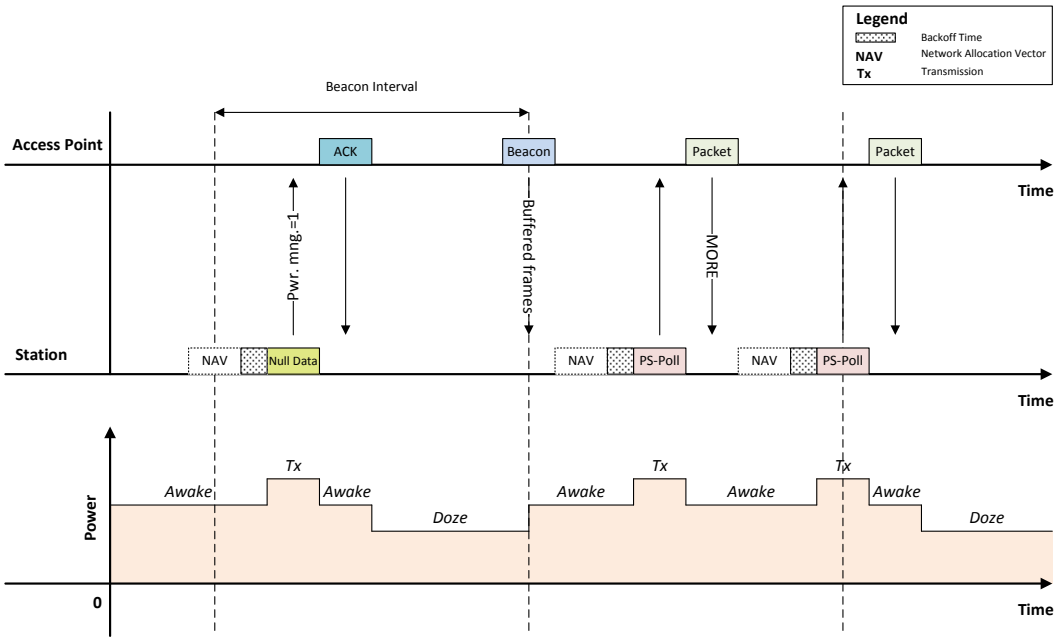


Figure 2.2: Simplified communication between an AP and a station in PSM. Based on [Tsao and Huang, 2011]

In this example, the station begins in the awake state since it is operating in No-PSM mode. In order to change its mode to Legacy-PSM, the station needs firstly to contend for the channel. The station verifies the *Network Allocation Vector (NAV)* that indicates when the channel becomes *idle*. If the channel is *idle*, the station selects a random backoff time and waits until it expires before sending the *NULL Data* frame. After receiving the *NULL Data* frame, the AP replies to the station by sending an *ACK* which informs the station that it is allowed to change its power save mode.

After changing the power mode and since there are no packets to receive, the station is able to sleep, reducing the energy consumption.

When the station is in the doze state, it can not send or receive any packet, therefore, to receive the *Beacon* frame sent by the AP, the station needs to wake up. The TIM, present in the *Beacon* frame, informs the station that the AP has buffered packets.

Therefore, the station needs to remain in the awake state to receive the pending data. As opposed, if the TIM informs that no packets are buffered for the station, the station can switch to the sleep state until the next *Beacon*. In order to receive the buffered packets, the station needs to poll every single packet by sending a *PS-Poll* frame after contending for the channel.

When the first *PS-Poll* frame is sent, the AP replies by sending one pending packet which contains the information that more packets are buffered in the AP. The station contends again for the channel and sends the *PS-Poll* frame to receive more packets. The AP sends another packet and informs that no more packets are pending. Since the AP does not have more pending packets, the station goes back to the doze state.

As seen in Figure 2.2, the station consumes more energy to send a packet than to receive. This happens because the station must amplify the signal to transmit a packet, in order to reach the AP.

The main two problems regarding energy efficiency of PSM are the need to poll every buffered packet and the need to wake up regularly to receive *Beacons*. When a station wakes up to receive a *Beacon* frame and no packets are buffered, energy is wasted. To overcome these limitations, the IEEE 802.11 standards introduced some mechanisms such as Automatic Power-Saving Delivery [IEEE, 2005] and Power Save Multi-Pool [IEEE, 2009]. However, the implementation of these mechanisms is not mandatory. Therefore, the PSM is the most implemented standard mechanism regarding energy consumption in IEEE 802.11. Moreover, in the last years, the Android devices started to implement some variations of the PSM mechanism, in order to overcome some of its limitations, as identified above.

2.4.2 Automatic Power-Saving Delivery (APSD)

The Automatic Power-Saving Delivery introduced in IEEE 802.11e [IEEE, 2005] defines two mechanisms, the Scheduled APSD (S-APSD), implemented in Enchanced Distributed Channel Access (EDCA) and also in HCF Controlled Channel Access (HCCA), and the Unscheduled APSD (U-APSD), implemented only in Enchanced Distributed Channel Access (EDCA). The APSD defines that a station can reserve a time period to exchange data traffic with an AP. This period, named Service Period (SP), allows the stations to reduce the number of collisions and signaling traffic, as opposed to the *PS-Poll* mechanism.

Following a similar approach to the stations operating in PSM, the stations that use the APSD mechanism must inform the AP to switch between the awake and sleep states.

When a station goes to sleep, the AP buffers the downlink frames.

In S-APSD, the AP periodically allocates SPs in established intervals. The station only wakes in these periods to transmit and receive packets without contending for the channel, leading to a lower energy consumption.

The difference between Scheduled APSD and Unscheduled APSD is that the AP does not allocate a Service Period for the stations using Unscheduled APSD in fixed intervals. A station in U-APSD is proactive and informs the AP in order to trigger a Service Period. However, the station must contend for the channel to inform the AP, thus wasting energy, as opposed to the S-APSD. The U-APSD brings advantages for example for Voice over IP (VoIP) applications. In VoIP applications, the data rate is similar in both directions, allowing a transmission of a downlink frame in a SP triggered by an uplink frame.

2.4.3 Power Save Multi-Poll (PSMP)

The PSMP mechanism defined in IEEE 802.11n was created to overcome the problem created by *PS-Poll* contentions made by several stations. This problem leads to a waste of resources of the AP and a waste in energy to the stations. In PSMP, the AP allocates a Downlink Transmission Time (DTT) and an Uplink Transmission Time (UTT) for each station. Therefore, when the AP sends the *Beacons*, the scheduled information is also sent. That scheduling is made considering different QoS requirements of the stations, in order to minimize *PS-Poll* contentions.

2.4.4 Adaptive Power Save Mode (Adaptive-PSM)

Recently, the Adaptive-PSM mode, also referenced as dynamic PSM, has been implemented in smartphones [Pyles et al., 2012] [Ding et al., 2012]. The idea behind the Adaptive-PSM mode is to switch between No-PSM and Legacy-PSM mode depending on the network traffic. This allows the device to avoid most of the delays presented in the Legacy-PSM mode due to the polling phase, however, it also reduces the time that the device may stay in the doze state, which leads to a lower energy efficiency. In fact, the Adaptive-PSM is a mode that aims to enhance the best of the No-PSM and Legacy-PSM mode by increasing the cost/benefit tradoff between the delay and performance.

Pyles et al. showed that the Adaptive-PSM mode implementations switch between No-PSM and Legacy-PSM modes by only using a threshold based on the network traffic amount [Pyles et al., 2012]. When the network traffic reaches the threshold value, the device switches to No-PSM mode. On the other hand, when the threshold is dropped,

the device switches to Legacy-PSM mode.

Moreover, the implementation of the Adaptive-PSM mode is different for each IEEE 802.11 driver implementations, since it is not defined in the standard. Some Adaptive-PSM implementations contain a pre-defined time (PSM timeout). The PSM timeout is the period that the device needs to wait before changing to the sleep state, right after a transmission or a reception period. When packets are exchanged between a station and an AP, if the Round Trip Time (RTT) is higher, the station might go into sleep and the packets will only be sent after the next *Beacon*. The PSM timeout is used to overcome this limitation.

2.5 Summary

This chapter presented the basic concepts of the IEEE 802.11 technology needed for this work.

The most important versions of the IEEE 802.11 standard were described and the main improvements of each version were presented. The description of IEEE 802.11 technology was focused on the MAC layer as well as the description of the most important power management modes defined in the IEEE 802.11 standard, namely the No-PSM, Legacy-PSM, the APSD and PSMP modes.

Furthermore, the Adaptive-PSM mode which has been widely implemented and used in mobile devices, such as the Android smartphones, was also described.

Chapter 3

Energy Efficiency in IEEE 802.11

This chapter introduces some optimization techniques regarding energy efficiency in the IEEE 802.11 technology. Firstly, mechanisms which aim to save energy in devices with IEEE 802.11 are summarized, followed by the presentation of the methodologies to assess energy consumption of the IEEE 802.11 interface. Finally, a comparison between the techniques described and an overall discussion is presented.

3.1 Optimization techniques

There are various proposals in the literature that aim to optimize the energy consumption in wireless devices. Watts2Share [Vergara and Nadjm-Tehrani, 2013a] is a hybrid solution proposed by Vergara and Nadjm-Tehrani for the Android platform. It uses heterogeneous networks (IEEE 802.11 and Third Generation of mobile phone standards and technology (3G)) to reduce the cellular energy consumption. This approach implies the use of various nodes that can have two roles, aggregator and coalition. The coalition nodes send the traffic to the aggregator node using the low energy radio interface (IEEE 802.11 interface). The aggregator node receives the traffic from the coalition nodes and transmits all the aggregate traffic via the high energy interface (3G). There is only one aggregator node at a time. The aggregator role changes accordingly to a role rotation algorithm that is very important to balance the energy consumption of the nodes, since the aggregator role is much more energy consuming than the coalition. The energy used by Watts2Share was estimated with EnergyBox [Vergara and Nadjm-Tehrani, 2013b] and the evaluation scenarios showed energy savings of 12% up to 59% when compared with a 3G basic scenario.

Vergara et al. proposed the Kernel Level Shaper (KLS) architecture [Vergara et al.,

2013] that uses the Cross-Layer Burst Buffering (CLBB) algorithm to perform background traffic aggregation and save energy in 3G. The CLBB algorithm performs data aggregation by taking into account the inactivity timers, the Radio Link Control (RLC) protocol thresholds and the maximum waiting time to transmit a packet (Tw). The traffic is intercepted in the networking stack with Netfilter and is categorized as small or large. Then, the packets go into two queues, one for small packets and the other for large packets. Later, the packets are transferred following the CLBB algorithm logic. When a station is in Pagin Channel (PCH) mode, the lowest power consumption mode in 3G, the CLBB waits until the Tw is about to expire to send packets, maximizing the time in this state. The evaluation tests performed with KLS showed that this architecture has a low overhead and can save energy between 7% and 58%. However, the tests with real time traffic showed that the mix of background traffic, in this approach, could lead to a bad performance in some applications. In applications like Skype, the delay produced by KLS leads to a significant increase in the number of packets (51 to 1145 in the tests made). Therefore, the identification of the background traffic is crucial for the success of this algorithm.

Pyles et al. argue that existing Adaptive-PSM mechanisms are based only in network thresholds, such as the aggregate traffic volume. This implies that important and high interactive traffic has the same weight as unimportant traffic. To overcome this limitation the Smart Adaptive PSM (SAPSM) [Pyles et al., 2012] was presented. The SAPSM prioritizes network traffic based on application priority (high or low). Only high traffic can switch the IEEE 802.11 interface to No-PSM mode. To do this, they implemented a kernel module and an Android service. The kernel module determines the priority of the packets. The Android application that runs as a service gets the total number of bytes transmitted and received by each application. With these metrics, the end-user can receive a suggestion with the priority of the applications based on an offline classifier. The offline classifier was done through a user study where the users interact with some specific applications. The applications have distinct network patterns and the users are allowed to set the applications priority in order to train the classifier. The evaluation results showed energy savings between 13% and 56%. However in some cases the classification of the applications is wrong. Additionally, the implementation of SAPSM is dependent upon the IEEE 802.11 driver.

The OPAMA [Bernardo et al., 2013] proposed by Bernardo et al. aims at saving energy while taking into account the end-user requirements. When a station receives continuous traffic, its wireless interface remains in the awake state for a long time and, therefore, the energy consumption is high. The OPAMA algorithm is an extension of

the standard Legacy-PSM mode and saves energy by keeping the packets buffered in the AP for longer. This leads to an increase of the time that a station is in the sleep state, reducing the energy consumption. When the station wakes up to listen to the *Beacon* frame, even if the AP has buffered packets, the OPAMA is able to hide this information from the station, which goes again to the sleep state. This decision is made by an algorithm that takes into account three parameters (station maximum allowed delay, α and β). The station maximum allowed delay is a parameter that is defined by the end-user. The α parameter defines a threshold for the number of video key frames to be queued, whereas β defines a threshold for the maximum number of aggregated frames to be queued. The OPAMA was developed and tested in a simulator and the results showed energy savings of 44% in a scenario with a low delay tolerance (100ms), when compared with the Legacy-PSM mode.

Pyles et al. proposed an implementation design named SiFi [Pyles et al., 2011], for Android devices. The SiFi aims to save energy for IEEE 802.11 when handling VoIP traffic. This proposal is based on the idea that in silence periods it is not needed to transmit packets and, therefore, the network interface can sleep. The implementation contains a Silence Classifier and Modeling and Prediction components, in order to predict the length of the silence period. The Silence Classifier determines when a silence occurs simultaneous in both directions. To determine if a silent event starts or ends, the Classifier verifies the average amplitude level of Real-Time Transport Protocol (RTP) packets and if this average is over a threshold, a silence period ends. On the other hand, when the threshold is dropped, a silence period starts. The Modeling component stores the cumulative length of silence periods and computes the Empirical Cumulative Distribution Function (ECDF). The Prediction module predicts the length of the sleeping period. In order to switch the IEEE 802.11 radio to sleep, the implementation contains a daemon (WiFi manager) that listens to the messages in a queue. When a message is received, the daemon sends a command to the "*wpa_supplicant*" in order to inform that the network interface must remain in Legacy-PSM mode for a given time in milliseconds. The "*wpa_supplicant*", which was modified, forces the IEEE 802.11 driver to switch to the Legacy-PSM mode and sleeps for a given time. Then, it wakes up and informs the IEEE 802.11 driver to switch again to the No-PSM mode. The evaluation of this design was conducted in a testbed and the results showed 40% of energy savings. Additionally, an evaluation of the application fidelity was made and showed minimal call quality degradation.

Ding et al. presented the Percy mechanism for IEEE 802.11 [Ding et al., 2012]. The Percy mechanism pretends to achieve on one hand the best of the Legacy-PSM

mode, maximizing energy saves, and, on the other hand, the best of the Adaptive-PSM mode, reducing the data transfer delay. The delay introduced in the Legacy-PSM mode occurs because the devices need to send a *PS-Poll* message to the AP for every buffered packet and because the device switches to the sleep state right after a transmission or reception period. The Percy implementation design uses a proxy behind an AP. This proxy intercepts the Transport Control Protocol (TCP) connections from the device and creates a new TCP connection to the end server. Afterwards, it transfers the data from the server and sends it back to the device. With the assumption that the Internet servers do not change, neither the mobile clients, an implementation of a local proxy leads to a small and constant delay to the client. Therefore, the PSM timeout can be small, reducing the overhead to enter in Legacy-PSM mode and allows the device to switch to the sleep state while the proxy transfers data. The validation tests were performed in three distinct devices including an Android smartphone and showed energy savings between 47% and 67% when compared to the different configurations of Adaptive-PSM mode.

Csernai and Gulyás proposed a mechanism for Android devices to reduce the energy consumption for video streaming applications, taking into account the quality of the video perceived by the end-users [Csernai and Gulyas, 2011]. This mechanism uses the U-APSD and was implemented as a video player plugin which contains two components: an estimation algorithm for the quality perceived by the end-users and a component that configures the U-APSD sleep schedule. This allows to dynamically adapt the frequency of time that the wireless interface wakes up to trigger a SP, reducing energy by taking into account the video quality. However, despite the fact that the authors argue that this mechanism is able to achieve energy savings between 20% and 30%, the energy consumption and also the quality of the video were not accurately measured.

Han et al. proposed the DozyAP which aims to reduce the energy consumption of the Android devices when sharing an Internet connection via the IEEE 802.11 interface [Han et al., 2012]. This mechanism works by implementing a protocol of sleep request and sleep response messages between the mobile devices. When a mobile device is acting as a hotspot, it can be in one of three different states: normal, pre-sleep or sleep. In the normal state, the mobile device has the wireless interface continuously awake allowing packets to be received or transmitted at any time. In the pre-sleep state, the wireless interface is idle and once it reaches a time threshold, the mobile device sends a sleep request message. If all the stations answer with a sleep response, the station acting as a hotspot is able to sleep. Otherwise, it goes back to the normal state. The sleep state represents the state when the wireless interface is turned off. The results showed

energy savings up to 33%, however, instead of setting the wireless interface into sleep to reduce the energy consumption, the wireless interface is turned off. This is done since a mechanism to fully control the wireless interface is not provided by Android devices.

The presented optimization techniques showed reductions in the energy consumption by introducing additional delay. However, a restricted group is focused in the energy consumed by the devices in the presence of Continuous Media Applications. Since the continuous traffic causes the IEEE 802.11 interface to stay awake for long time periods, as will be shown later, the proposal of optimization techniques, which aim to reduce the energy consumption of the IEEE 802.11 interface in the presence of Continuous Media Applications, must be a concern.

3.2 Measurement Technologies

The study of the real impact in energy consumption by the different optimization techniques is very important. Therefore, some frameworks have been proposed in order to assess the energy consumption in real devices.

The EnergyBox proposed by Vergara et al. [Vergara and Nadjm-Tehrani, 2013b] is a tool that allows the end-users to obtain an accurate estimation of the energy consumption in 3G and IEEE 802.11. The inputs of EnergyBox are the application data traces and some specific configuration parameters of the mobile device. The output corresponds to the phone states over time. The energy consumption is calculated by matching these states with the corresponding power levels. The results of this tool have been validated through comparisons with real measurements and showed an average accuracy of 98% for estimated energy consumption in 3G and IEEE 802.11.

Bernardo et al. proposed an empirical methodology for assessing the energy consumption [Bernardo et al., 2011] in an experimental testbed. The proposed methodology contains four main components to assess the energy, namely a digital multimeter, a controller machine, an end-user device and an Universal Serial Bus (USB) network interface. The digital multimeter is responsible for a high precision measurement and is connected to the USB network interface and the controller machine. Therefore, the energy consumption of the USB network interface is assessed by the digital multimeter and the results are stored in the controller machine. The controller machine is able to control and send commands to the digital multimeter, in order to perform a set of repeatable experiments created by test scripts. The USB network interface allows the mechanism to be independent of the interface. However, the connection of the USB network interface directly to the end-user device has an impact in the consumed voltage. An

Alternative Current Power (AC-Power) that provides a stable voltage to the the system was used to annul the voltage drain. Moreover, some tests were made to evaluate IEEE 802.11 2.4 GHz, 5 GHz and IEEE 802.18e (Worldwide Interoperability for Microwave Access (WiMAX)). These tests showed the energy consumed in each technology state, the impact of the packets size and the impact of the transmission rate.

Bernardo et al. also proposed a methodology to assess video energy consumption [Bernardo and Curado, 2012]. This methodology uses the setup presented in [Bernardo et al., 2011] and extends it to allow the video quality assessment. The methodology provides a mechanism to evaluate the video transmission under different network conditions and obtain Quality of Experience (QoE) metrics. To generate the video traffic the Evalvid framework was used and the link emulator was provided by the Dummynet-enabled bridge.

Rice et al. presented a methodology to assess the energy consumption of IEEE 802.11 in Android devices [Rice and Hay, 2010]. To measure the energy consumption, the battery was replaced by a printed plastic one. The plastic battery is used as a support to connect a high precision measurement resistor in series, between the battery and its connection into the smartphone. This allows the voltage measure of the phone battery with a board. In order to perform the assessment tests, the overall power consumption was separated by the components. Therefore, most of the device components were switched off. To overcome the Central Proccessing Unit (CPU) multitasking from allowing other processes to consume resources, a low-priority process in background is running to consume all spare CPU cycles. Moreover, the energy traces are synchronized with the recorded events. This is done by creating a predetermined pattern which permits to correlate the events. To automate the test process, some test scripts were created. Therefore, when a test is about to start, the device loads a script which makes a baseline power calibration. Some of the smartphone components need to be turned on, in order to conduct some tests. The calibration mechanism allows the removal of the power consumption wasted by this components, in the rest of the test.

The methodologies to assess the energy consumption of IEEE 802.11 presented are mostly performed in testbed. In fact, the use of a measurement testbed setup, with high precision hardware, allows to understand the real energy consumption of IEEE 802.11 interface.

3.3 Comparison

This section summarizes the chapter by performing a comparison between the different energy consumption optimization proposals and the methodologies to assess the energy consumption.

Table 3.1 presents the main parameters regarding the characteristics of the proposed mechanisms for energy consumption optimization. These parameters are described bellow:

- **Implementation**: Compares the mechanisms concerning its implementations (simulation or real devices);
- **Energy assessment type**: The energy assessment can be conducted through simulation or empirical evaluation;
- **Network technology**: The network technology used in the methodology;
- **Traffic classification**: Compares the methodologies used to classify the traffic. Additionally, the classification type is presented;
- **End-users expectations**: Informs if the mechanism takes into account the end-user expectations, namely the maximum allowed delay tolerated or the quality expected.

After analyzing the various proposed mechanisms, it is possible to understand the approaches that aim at energy efficiency in IEEE 802.11. Most of the methods have the goal of keeping as long as possible the stations in the sleep state. The increase of time in this state is normally performed through packet aggregation or by keeping the packets in the AP queues for longer. Moreover, all the proposed mechanisms showed a reduction in the energy consumed. However, these mechanisms also introduce some delay and overhead due to the increase of time that the packets stay in queues, before being transmitted. Another problem regarding the proposed mechanisms in the literature is that some of them are evaluated by simulation, which means that the results in a real environment could be different. On the other hand, some mechanisms, even when evaluated in testbed, do not present an accurate performance analysis.

Furthermore, most of the methodologies intend to save energy, but do not take into account the expectations by the end-user.

	Implementation	Energy Assessment Type	Network Technology	Traffic Classification	End-user Expectations
Watts2Share [Vergara and Nadjm-Tehrani, 2013a]	Android devices	Simulation	IEEE 802.11 and 3G	No	No
SAPSM [Pyles et al., 2012]	Android device	Empirical	IEEE 802.11	Important and Unimportant	No
Kernel Level [Vergara et al., 2013]	Android device	Simulation	3G	Small and Large	No
OPAMA [Bernardo et al., 2013]	Simulator	Simulation	IEEE 802.11	No	Yes
SiFi [Pyles et al., 2011]	Android device	Empirical	IEEE 802.11	No	Yes
Percy [Ding et al., 2012]	Smartphone	Empirical	IEEE 802.11	No	No
U-APSD - Video streaming [Csernai and Gulyas, 2011]	Android device	Empirical	IEEE 802.11	No	Yes
DozyAP [Han et al., 2012]	Android device	Empirical	IEEE 802.11	No	No

Table 3.1: Comparison between energy consumption optimization mechanisms.

Table 3.2 summarizes the differences between the energy consumption assessment methodologies regarding the following parameters:

- **Assessment type**: Distinguishes the assessment methodologies by its type, simulation or testbed;
- **Platform**: Specifies the platform for which the methodology was developed;
- **Network interface**: The network interface that can be assessed;
- **High precision hardware**: Informs if the methodology uses a high precision hardware to measure the energy consumption.

Concerning the energy consumption assessment, it is very important to evaluate the energy in a testbed environment. The energy assessment in testbed makes it possible to understand the real impact in energy consumption of the proposed mechanisms. Moreover, the energy assessment with high precision hardware is also an important goal. In fact, network interfaces, such as IEEE 802.11, could have slight variations and, therefore, high precision results must be performed to verify the true impact of these interfaces.

	Assessment Type	**Platform**	**Network Interface**	**High Precision Hardware**
EnergyBox [Vergara and Nadjm-Tehrani, 2013b]	Simulation	Android	IEEE 802.11 and 3G	No
Measurements in a Cloud Computing Wireless [Bernardo et al., 2011]	Testbed	Non-specific	Independent	Yes
Video transmission energy consumption and quality [Bernardo and Curado, 2012]	Testbed	Non-specific	Independent	Yes
Mobile phone energy consumption for IEEE 802.11 [Rice and Hay, 2010]	Testbed	Android	IEEE 802.11	Yes

Table 3.2: Comparison between energy consumption assessment methodologies.

3.4 Summary

In this chapter some optimization techniques presented in the literature, which aim to reduce the energy consumption of the IEEE 802.11 interface, were described. Moreover, some methodologies to assess the energy consumed by the IEEE 802.11 interface were also presented. A discussion about the optimization techniques as done by comparing the mechanisms. The comparison showed that the presented mechanisms reduce the energy consumption by keeping the packets in queues for longer and by performing aggregation. This allows the wireless interface to stay in the sleep state for longer periods with the cost of adding delay. Moreover, the comparison between the methodologies to assess the energy consumption of IEEE 802.11 showed an importance to evaluate the energy consumption in testbed.

Chapter 4

IEEE 802.11 Interface Characterization

This chapter describes the testbed setup and the performed tests which aim to characterize the energy consumption of the IEEE 802.11 interface in Android devices. The testbed was created to perform the tests in a real environment allowing to understand the impact in the energy consumption of some Android components, such as the display and IEEE 802.11 interface.

Section 4.1 presents the experimental environment followed by the description of the test scenarios in Section 4.2 and the analysis of the results in Section 4.3. Finally, a summary of the chapter is presented in Section 4.4.

4.1 Experimental Environment Overview

This section presents an overview of the experimental environment setup and is divided in three subsections. First, the energy measurement testbed is described followed by the mobile phone setup. Finally, the network setup is presented.

4.1.1 Energy Measurement Testbed

The energy measurement setup aims at addressing key features namely, the testbed assessment, the evaluation of the smartphone energy consumption and the use of a high precision measurement. Many assessment methodologies are based on simulations, which do not represent the true impact of the energy consumed in the real systems. Since the energy consumed will be assessed on a smartphone, it is necessary to create a

mechanism that allows the smartphone energy consumption assessment. Moreover, the energy consumed along the time by the smartphone has slight variations. Therefore, it is important to use hardware that can measure the energy with high precision, i.e. is able to support multiple samples per second.

Figure 4.1 represents a scheme of the energy measurement testbed which includes four components to access the energy consumption, namely the Digital Multimeter, the "Controller Machine", the Mobile Device and the Direct Power Supply.

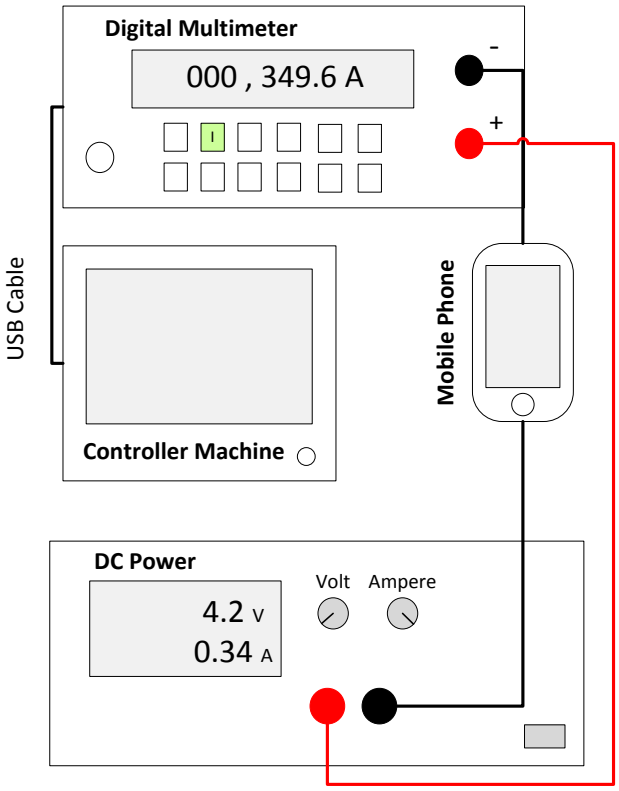


Figure 4.1: Energy measurement setup. Based on [Bernardo et al., 2011]

The Digital Multimeter (Rigol DM3061) supports a sampling rate of 1000 samples per second, being the component responsible to measure the energy consumption with a high precision. The Digital Multimeter is connected with an USB cable to the "Controller Machine", which is a netbook that controls and manages the Digital Multimeter by using Standard Commands for Programmable Instruments (SCPI) commands. The "Controller Machine" is then used to run and save a set of repeatable tests. The Digital Multimeter is also connected to the Mobile Phone and a Direct Power Supply, that is

able to generate a continuous stable Voltage (V). For the proper function of the Mobile Phone, the Direct Power Supply is configured to provide 4.2 V. This allows the unit that controls the phone charging process to operate in "*Factory Mode*", preventing that the mobile device turns off by reporting that the battery is not charged.

The energy measurement testbed was based on the work performed by Bernardo et al. [Bernardo et al., 2011] and adapted according to the needs of this work, namely the modification of the testbed to measure the energy consumption of the Android device, instead of an USB network interface.

4.1.2 Mobile Phone Setup

The energy consumption assessment of the smartphone is based on Pyles et al. work [Rice and Hay, 2010] and is granted by a battery holder that replaces the battery. However, as opposed to the Pyles et al. proposal, where a resistor in series is placed between the holder cables and the battery, leading to a dependency of the battery pattern discharge, the battery holder in this work contains a connector that allows the direct link of the Mobile Phone to the Direct Power Supply and to the Digital Multimeter. Table 4.1 presents the main specifications of the used device:

Table 4.1: Device main specifications.

Features	**Model**	LG Optimus 2X
	Chipset	Nvidia Tegra 2 AP20H
	CPU	Dual-core 1Ghz Cortex-A9
	RAM	512 MB
	WLAN	IEEE 802.11 b/g/n, DLNA, WiFi hotspot
Setup	**OS**	Android 4.2.2
	ROM	CM 10.1-20131006-Nigtly-p990
	Kernel	Kowalski-kernel-100p5-oldbl
	Bootloader	old bootloader
Display	**Type**	IPS LCD capacitive touchscreen, 16M colors
	Size	480 x 800 pixels, 4.0 inches (233 ppi pixel density)

In order to conduct a set of repeatable tests in the testbed setup, various scripts were created. However, one of the initial problems was that the Android OS only allows the display to be awake for a maximum of 30 minutes. This would be a critical problem since a sequence of tests could not be performed, under the same conditions, in a time period greater than 30 minutes. An application was developed to overcome this limitation. This application contains a switch button that allows the user to lock or unlock the brightness of the display. When the switch button is on, a service is started and acquires a wake

lock of the display brightness, ensuring that the screen does not turn off.

4.1.3 Network Setup

This subsection presents the network setup used in the testbed. Figure 4.2 illustrates the IEEE 802.11 architecture composed by a Core Network and an Access Network.

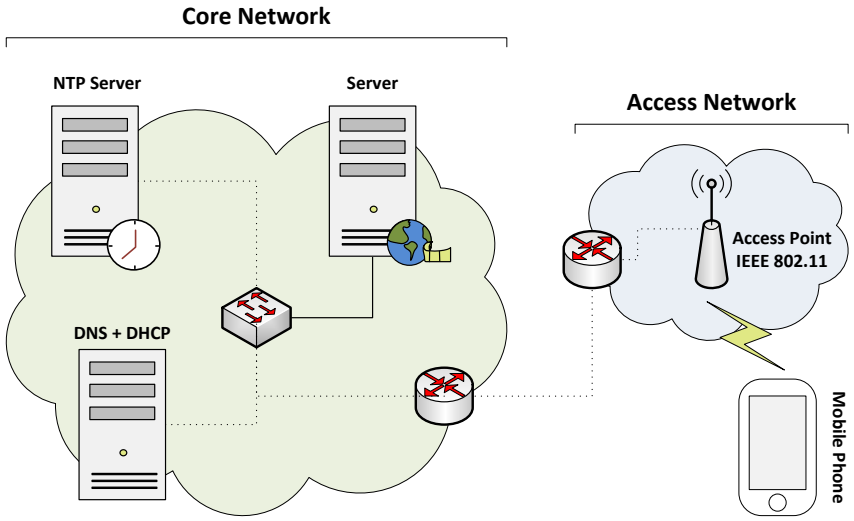


Figure 4.2: Network setup.

The "*Mobile Node*" represents the Android device specified in Subsection 4.1.2 that is connected to a commercial "*Access Point*", a Cisco Linksys E4200. The main configurations of the AP are presented in Table 4.2.

Table 4.2: Commercial AP main configurations

Parameter	**Value**
Beacon interval	100 ms
DTIM interval	3
Buffer size	1024 packets

The *Beacon* interval defines the frequency interval to transmit each *Beacon* while the *DTIM* interval indicates which multiples of *Beacon* frames contains the TIM. Therefore, with this configuration, the AP only sends information of pending packets in the third *Beacon* frame (i.e. every 300 ms). Concerning the size of the queues, the AP allows to buffer 1024 packets.

The Core Network runs in a HP ProLiant DL320 G5p server and is composed by 3 virtualized components running in Debian 7.0: the "*Server*", the 2*NTP Server*" and

the "*DNS+DHCP*".

The preliminary tests performed in the first semester using the iperf tool [Iperf, 2014] do not provided enough metrics to evaluate the results (e.g. delay). Therefore, the Distributed Internet Traffic Generator (D-ITG) version 2.8.1 [Botta et al., 2012] was cross-compiled for Android. Since the D-ITG tool was not currently supported in Android, its cross-compilation also led to the replacement of some source code to make it possible to use this tool. Therefore, a patch was developed to fix the "*pthread*" lib functions that are not supported by the Android bionic. This patch contains a signal mechanism to send and handle a user signal which replaces the unsupported source code.

All the tests containing traffic were performed by using the D-ITG tool to generate traffic from the "*Server*" to the "*Mobile Node*".

The "*Server*" and the "*Mobile Node*" were synchronized via the "*NTP Server*" using the Network Time Protocol daemon (ntpd) and the ClockSynch application, respectively. The ClockSynch application was used through Android testing, which allows the interaction with the application via scripting. Furthermore, this application allowed the synchronization of the Android device without incurring with extra energy costs.

4.2 Experimental Evaluation

This section describes the experimental setup for the tests performed. The objectives are presented first, followed by the description of the scenarios and the configurations for each test.

4.2.1 Objectives

The main goal of the experimental evaluation is to study the real impact of the IEEE 802.11 implementation in Android, regarding its energy consumption and network performance.

The mobile device allows the usage of three different power management modes of IEEE 802.11: No-PSM, Adaptive-PSM and Legacy-PSM.

When the device is operating in No-PSM mode, the wireless interface is always awake listening to the channel. This allows higher throughputs and lower delays, however, this mode is responsible for the highest energy consumption.

The Adaptive-PSM switches between No-PSM and Legacy-PSM (described next) depending on the network traffic, allowing the IEEE 802.11 interface to stay awake only if traffic is present.

When the Legacy-PSM is employed, the standard defines that the AP buffers the downlink packets when the device is sleeping. The device only wakes up to receive the *Beacon* frames and poll the buffered packets. If the *Beacon* frame informs that the AP has pending packets, the wireless interface wakes up to receive the packets. This is the lowest energy consumption mode, however, it introduces additional delays.

A set of scenarios was created to analyze the energy consumption with the different IEEE 802.11 power management modes. Additionally, the energy consumption by the device display is also presented.

4.2.2 Scenarios and Configurations

The performed tests were conducted in three different scenarios. The first scenario aims to study the energy consumed by the device display. The second scenario investigates the energy consumption in the IEEE 802.11 interface states. The third scenario analyzes the energy consumed by the device, in the presence of a Continuous Media Application, using the different power management modes of IEEE 802.11.

In all the scenarios, the tests performed have no extra applications running in background and the tests concerning the energy consumed by IEEE 802.11 technology have the display brightness at 100%. Each test has a duration of 60 seconds and was done with a rate of 1000 samples per second. All the results are based on 20 runs and are exhibited in the charts with a confidence interval of 95% assuming that the measurements follow a normal distribution.

4.2.2.1 Display Brightness Scenario

This scenario aims at testing the display brightness impact in the energy consumption of the smartphone. In this scenario the IEEE 802.11 radio is always off (wireless interface disconnected) and only the display brightness is increased by steps of 25%, between 0% and 100%. This scenario will allow to remove the baseline energy of the mobile device in the next scenarios, in order to understand the real impact of the energy consumed only by the IEEE 802.11 interface.

4.2.2.2 IEEE 802.11 States Scenario

In this scenario the IEEE 802.11 interface states will be analyzed regarding their energy consumption. The IEEE 802.11 interface can be in two states, disconnected or connected. In the disconnected state the device radio is off. On the other hand, in the connected state, the radio is on and the device is associated with a network.

When the wireless interface is connected, it can be in the awake state or in the sleep state (also called by doze state). When in the awake state, the device is able to receive or send packets. As opposed, in the doze state the IEEE 802.11 interface is sleeping. Therefore, in this scenario two tests were performed to understand what is the ratio between the energy consumption when the wireless interface is awake and when it is in sleep.

4.2.2.3 Continuous media application scenario

This scenario aims to verify the impact in the energy consumption and network performance of the No-PSM, Adaptive-PSM and Legacy-PSM modes in the presence of continuous UDP traffic. Firstly, the packet size impact will be analyzed followed by the study of the impact of the transmission rate on energy consumption.

As opposed to the above scenarios, in this scenario each test has a duration of 80 seconds. However, the D-ITG runs only for 60 seconds, to send traffic between the 10 and 70 seconds of the total test. This allows the test to have two zones called by heating and cooling zone, respectively at the beginning and at the end of the sent traffic. At the end of each test, the energy consumption is parsed to include only the time where the traffic is sent (60 seconds).

4.3 Results and Analysis

In this section the results obtained in the different scenarios are presented. Therefore, this section is divided in three subsections, representing each scenario, and a subsection with the discussion of the obtained results. Subsection 4.3.1 presents an analysis of the display brightness results and is followed by the analysis of the IEEE 802.11 states, in Subsection 4.3.2. The tests with network data traffic are analyzed in the Subsection 4.3.3. Finally, a discussion about the results is performed in Subsection 4.3.4.

4.3.1 Display Brightness

In order to verify the impact of the display brightness in the energy consumption of the smartphone, a set of tests was performed.

Figure 4.3 presents the average electric power (miliwatt), in the y-axis, consumed by different levels of the display brightness, in the x-axis.

As showed, the display brightness is responsible for a large part of the energy consumed by the smartphone. In fact, the brightness at 100% can lead to an overall energy

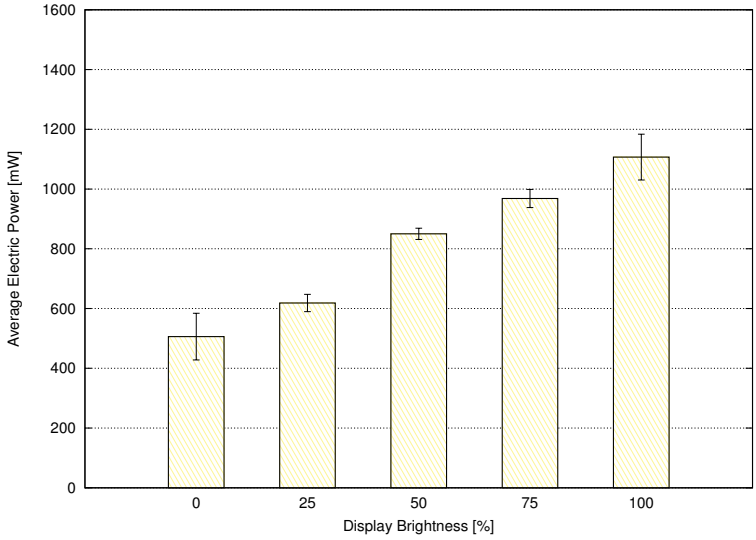


Figure 4.3: Average power consumed with different display brightness.

consumption higher than 50%, compared with the brightness at 0%. Even though the energy consumed by the device display is still different from each mobile phone, with different screens, others like Pyles et al. [Pyles et al., 2011] and Kennedy et al. [Kennedy et al., 2011] already experienced an higher energy consumption of the display.

Furthermore, the energy consumption of the smartphone is proportional to the display brightness. The average power consumed in 1 second, varies between the 400mW, when the display brightness is 0%, and the 1200mW, with the display brightness at 100%.

4.3.2 IEEE 802.11 Power Management States

Figure 4.4 shows the power consumed by the smartphone with IEEE 802.11 interface connected (y-axis) in the two different states, awake and sleep (x-axis). The power consumed with the wireless interface disconnected is displayed, in order to have a base for comparison and was provided by the previous scenario, since it is the same as the test with the display brightness at 100%.

These results show that the smartphone with the network interface awake consumes an average power over 1400mW, more 29.77% when compared with the network interface disconnected. On the other hand, the average power consumed by the IEEE 802.11 interface in the sleep state is only 4.96% above the IEEE 802.11 interface disconnected.

The real impact in the energy consumption by the wireless interface of the mobile

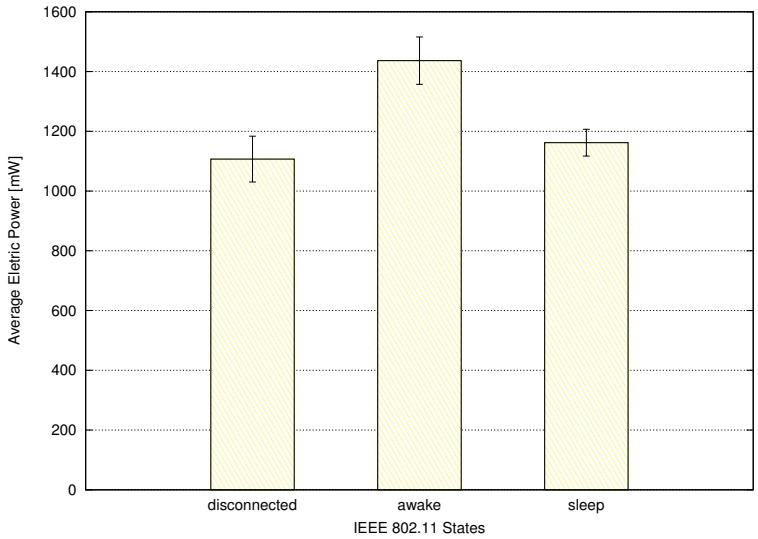


Figure 4.4: Average power consumption in different IEEE 802.11 states.

device can be obtained by removing the baseline energy of the device (i.e. when the device is disconnected and the screen brightness is at 100%). Therefore, it is possible to obtain the ratio between the energy consumption of the wireless interface awake and in sleep which is more than 6 times. Other works in the literature have already noticed a big difference in the energy consumption of the network interface, like Pyles et al. [Pyles et al., 2011] and Kennedy et al. [Kennedy et al., 2011].

4.3.3 Continuous Media Application

This subsection presents the No-PSM, Adaptive-PSM and Legacy-PSM modes performance in the presence of a Continuous Media Application. First, the impact of the packet size will be discussed, followed by the analysis of the transmission rate impact.

4.3.3.1 Impact of packet size

To study the impact of the packet size, the transmission rate was fixed at 100 packets per second and the packet size was changed for each test. By performing some preliminary tests, the 100 packets per second value for the transmission rate was chosen since it represents a rate where all the studied power modes depicts a normal behavior. Nonetheless, this rate could be set with other fixed values. Since the only change in each test will be the packet size, the rate will not change the results trend.

Figure 4.5 presents the total energy consumed by the wireless interface in Joule (y-axis) for No-PSM, Legacy-PSM and Adaptive-PSM modes. The packet size was set to 200, 600, 1000 and 1400 bytes (x-axis). From now on, the results concerning the total energy consumption by the mobile device are presented without the baseline energy (i.e. the wireless interface disconnected and the screen brightness at 100%). This allows to understand the real impact of the power modes in the energy consumption of the wireless interface.

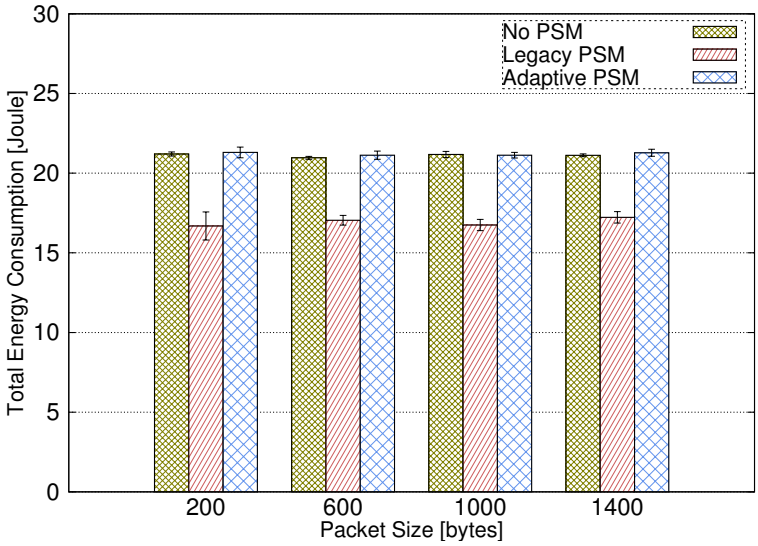


Figure 4.5: Total energy consumed by the IEEE 802.11 interface with different packet sizes.

The results showed a clear benefit of using higher packet sizes, since the impact of increasing the packet size is negligible in the energy consumption of the wireless interface, as already verified in other works performed by Trestian et al. [Trestian et al., 2012] and Bernardo et al. [Bernardo et al., 2011]. Therefore, a higher packet size leads to a lower energy cost to transmit one byte.

Concerning the behavior of the three power management modes, it is clear that the Legacy-PSM is the one that requires the lowest energy to receive the same amount of packets, between 18% and 21% less. Moreover, the depicted results also showed the gap of the Adaptive-PSM in the presence of a Continuous Media Application. Since the traffic is sent continuously and is always present in this type of applications, the Adaptive-PSM can not set the wireless interface to sleep. This leads to a power consumption similar to the one when the No-PSM is employed.

Even though the Legacy-PSM is the mode that consumes the lowest energy, it is

also important to understand the impact of the modes by analyzing the Quality of Service (QoS) metrics.

Figure 4.6 depicts the one way delay, in milliseconds (y-axis), for each packet size (x-axis) when using the three power management modes.

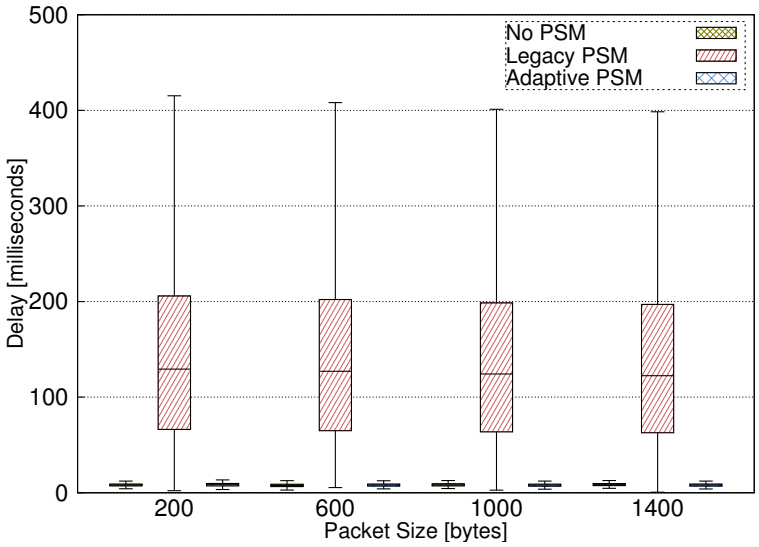


Figure 4.6: One way delay for all IEEE 802.11 power management modes.

The results show that the delay introduced by the Legacy-PSM is much higher than the ones with No-PSM and Adaptive-PSM. In fact, the delay median when the interface operates in Legacy-PSM is around 125 ms and the maximum delay is up the 400 ms.

To understand better the impact of the delay introduced by the Legacy-PSM, Figure 4.7 zooms only the one way delay presented in No-PSM and Adaptive-PSM.

The No-PSM and Adaptive-PSM modes have delays only with a median between 7 and 9 ms and a maximum that does not exceed 14 ms. Therefore, this shows that the trade-off between the energy consumption and the delay introduced by the Legacy-PSM is not good. This happens due to the need to send a PS-Poll frame for each pending packet, when the Legacy-PSM mode is used. This polling phase also leads to some packet losses, but always lower than 0.2%.

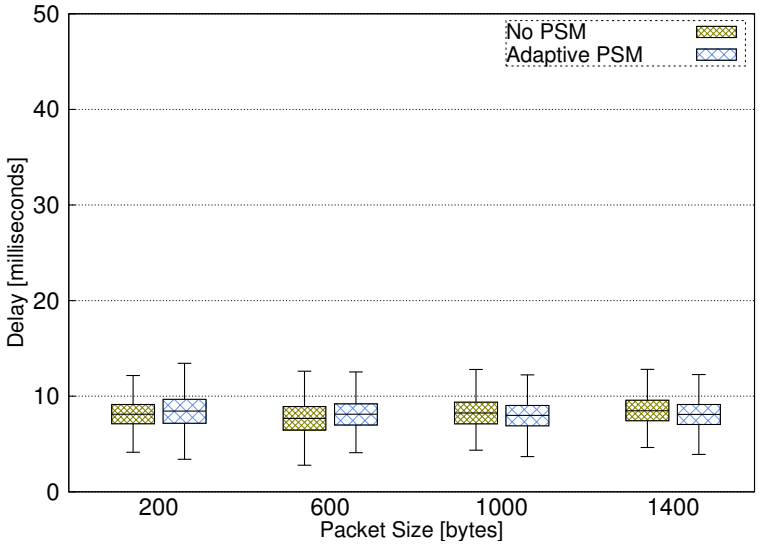


Figure 4.7: One way delay for No-PSM and Adaptive-PSM.

Additionally, it shows that the delay in No-PSM and Adaptive-PSM is similar and once more, the packet size does not have any effect in the modes performance.

4.3.3.2 Impact of transmission rate

The impact of the transmission rate will be evaluated by setting a fixed packet size of 1000 bytes and varying the transmission rate between 50 and 250 Mbps. The packet size was fixed with 1000 bytes since this value can represent, for example, a video frame, which is used in Continuous Media Applications, such as the video streaming [Trestian et al., 2012].

Figure 4.8 depicts the total energy consumption (y-axis), in Joules, of the three IEEE 802.11 power management modes while receiving continuous UDP traffic. The continuous traffic is sent in each test with a different packet rate (x-axis).

The results showed a linear relationship between the energy consumption of the device's wireless interface and the packet rate, when using No-PSM and Adaptive-PSM modes. On the other hand, when operating in Legacy-PSM, this relation is only visible for rates lower then 190 Mbps. The energy consumption of the device in Legacy-PSM mode above this rate, even though it has some fluctuations, remains almost constant. Moreover, the Legacy-PSM mode only saves energy in the studied rates above 150 Mbps when compared with the other modes. Therefore, it is possible to conclude that the packet rate has a strong impact in Legacy-PSM.

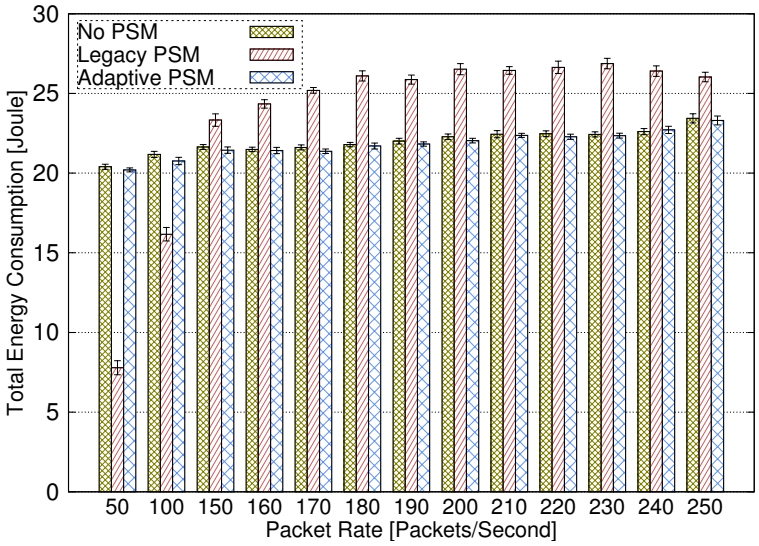


Figure 4.8: Total energy consumed by the IEEE 802.11 interface with different transmission rates.

To investigate the impact of the packet rate in this mode, Figure 4.9 shows the one way delay (y-axis) introduced for each transmission rate (x-axis).

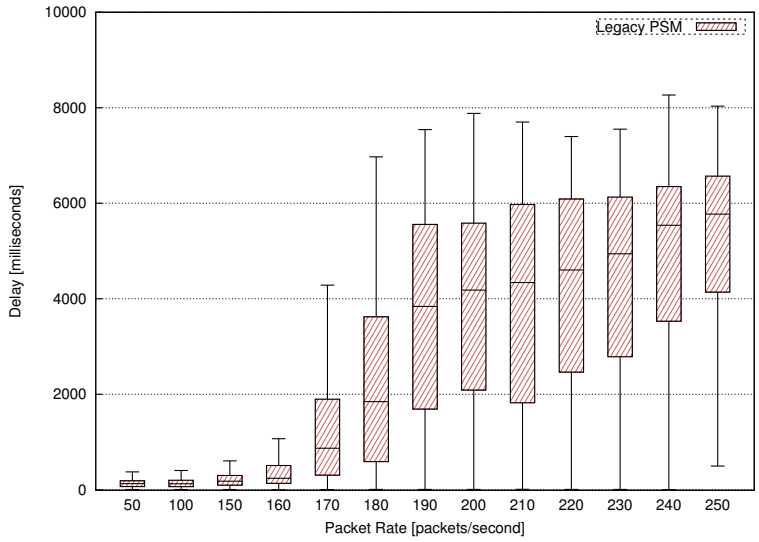


Figure 4.9: One way delay for Legacy-PSM with different transmission rates.

As well as the energy consumed, when operating in Legacy-PSM, the one way delay is also highly affected by the increase of the transmission rates.

The results show a high delay introduced by Legacy-PSM which with the 160 Mbps

has already a maximum delay above 1 second. Furthermore, the existent median delays above this rate are unacceptable, reaching a maximum of almost 6 seconds. As opposed, the delays presented with No-PSM and Adaptive-PSM are much lower.

Figure 4.10 depicts the one way delay (y-axis) for No-PSM and Adaptive-PSM. The delays observed showed a median between 7 and 9 ms, while the maximum delay is always lower then 14 ms. These values meet the results presented in the impact rate evaluation, leading to the conclusion that the transmission rate does not affect the QoS performance of the No-PSM and Adaptive-PSM in the studied rates and scenarios.

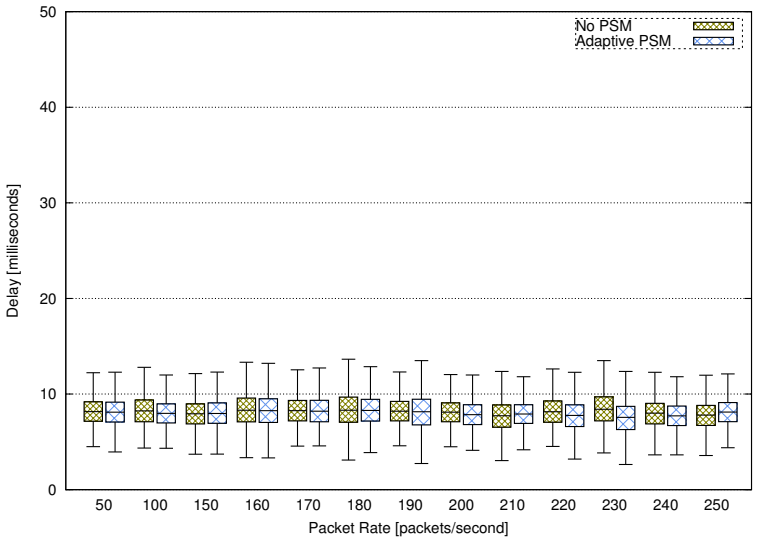


Figure 4.10: One way delay for No-PSM and Adaptive-PSM.

In addition to the one way delay, it is important to complement the QoS analysis by observing if packet loss occurs when the device operates in Legacy-PSM.

Figure 4.11 shows the Legacy-PSM percentage of packet loss (y-axis) for each transmission rate (x-axis).

The results show that Legacy-PSM also introduces higher packet losses. In fact, above the 180 Mbps, the percentage of packet loss is unacceptable leading to a decrease of the QoS. This behavior is expected due to the Legacy-PSM design implementation. As defined in the IEEE 802.11 standard, a station operating in PSM must wake up to receive each *Beacon* frame. If the frame indicates that there is pending packets in the AP, the station must stay awake and send a *PS-Poll* frame to receive each packet. This polling phase decreases the throughput of the transmitted frames, leading to higher delays and higher energy consumption. Moreover, if the delays became to high, the AP

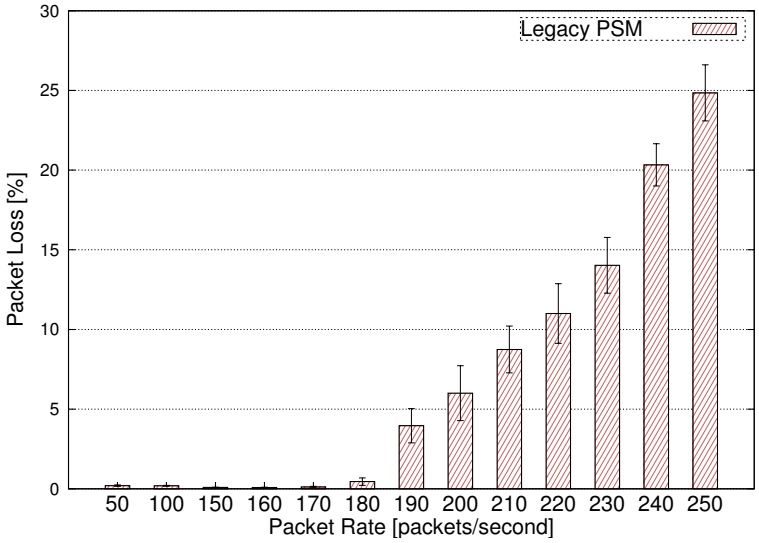


Figure 4.11: Packet loss introduced by Legacy-PSM with different transmission rates.

is forced to purge the old packets in the pending queues due to time constraints or lack of space.

4.3.4 Discussion

The analysis of the obtained results conducted in three distinct scenarios, namely the energy consumption of the device display, the energy in the different states of the wireless interface and the characterization of the IEEE 802.11 power management modes in the presence of continuous traffic, allowed to conclude that:

- **Energy consumption by the display:** The depicted results showed that the device display is a high energy component and is responsible for a significant part of the device energy consumption. Since the development of mechanisms to reduce the display energy consumption are beyond the scope of this work, the presented tests were conducted to characterize the energy consumption of the device. Moreover, these results also allowed to realize what is the baseline energy consumed by the device which was used to understand the real impact in the energy consumption of other components (e.g. wireless interface);

- **Energy consumed by the IEEE 802.11 interface in sleep:** When the IEEE 802.11 interface is sleeping, most of the radio components can be switched off. This leads to a very low energy consumption. The conducted tests revealed that, while in sleep, the wireless interface energy consumption is quite similar to the energy consumed with the wireless interface disconnected, as opposed to the interface in the awake state. This highlights the importance of developing new mechanisms that keep the wireless interface in sleep longer, to reduce the devices energy consumption;

- **Continuous Media Applications:** Despite of the low energy consumption of the IEEE 802.11 interface in the sleep state, the current implemented power save modes, in Android devices, are not able to keep the wireless interface in sleep for longer, in the presence of continuous traffic. As showed by the tests performed, the Adaptive-PSM mode, in the presence of Continuous Media Applications, has a similar behavior when compared with the No-PSM mode. This happens since traffic is always present in this type of applications, forcing the Adaptive-PSM mode to set the wireless interface to be continuously awake, as in No-PSM mode. On the other hand, the Legacy-PSM mode is able to reduce some energy consumption, however, this is only true for the lowest rates studied. Furthermore, due to the polling phase performed when a station is operating in Legacy-PSM mode, the throughput is decreased and, as a result, the delays can reach unacceptable values as well as high packet losses. Therefore, it becomes essential to develop mechanisms which are able to balance the trade-off between the energy consumed by the devices and the end-user expectations, in the presence of Continuous Media Applications;

- **Framework to control IEEE 802.11:** In order to develop mechanisms to reduce the energy consumption of the Android devices, in the presence of Continuous Media Applications, it is crucial to control the IEEE 802.11 interface modes and states. However, the Android devices do not provide full control of the IEEE 802.11 technology in higher layers (e.g. applications) neither provide an interface to force the wireless card into sleep. Therefore, developing a framework which allows the full control of the IEEE 802.11 interface becomes a challenge. This framework will allow the propose and validation of mechanisms capable of saving energy for Continuous Media applications by managing the wireless interface to stay in sleep more time.

4.4 Summary

This chapter presented the IEEE 802.11 technology energy consumption characterization in an Android device. In order to fully study the IEEE 802.11 technology, a testbed which contains an energy measurement setup was developed and described.

By conducting a set of tests in three different scenarios, it was possible to understand that the implemented IEEE 802.11 power management modes, in Android devices, do not achieve a good trade-off between the energy consumption and the QoS, while receiving continuous traffic. In fact, the Legacy-PSM mode introduces unacceptable delays and losses, while the Adaptive-PSM mode is not able to reduce the energy consumption. Therefore, the development of a framework which enables the full control of the IEEE 802.11 interface becomes essential to support mechanisms capable of reducing energy in the presence of Continuous Media Applications, since, the Android devices do not provide these control.

Chapter 5

EXPoSE Framework

This chapter presents the Android framework for Extending Power Saving control to End-users (EXPoSE) developed in this thesis. The objectives of the proposed framework are described first, followed by a discussion about the architecture and specification. The implementation details of the framework are presented next and, finally, the results of the framework validation, in a real testbed, are analyzed.

5.1 Objectives

The EXPoSE framework was created to overcome the limitations of the existing IEEE 802.11 power saving management modes, implemented in the Android devices, presented by the results discussed in Chapter 4.

The most deployed mechanism regarding power saving in IEEE 802.11 is the Legacy-PSM. More recently, the mobile devices have been implementing a variation of this mechanism, called Adaptive-PSM mode. The Adaptive-PSM differs from the Legacy-PSM, as it switches between the sleep and awake states depending on the network traffic, avoiding the polling phase. However, as previously shown in Chapter 4, both Legacy and Adaptive power modes do not show improvements in the presence of Continuous Media Applications. Moreover, these power saving techniques do not take into account the end-user expectations. Therefore, the first goal of the EXPoSE framework is to provide a mechanism which allows the control of the IEEE 802.11 interface power modes by the end-users, in Android devices.

The second goal of the EXPoSE framework is to extend the default interface of the IEEE 802.11 technology in Android devices, to allow the wireless interface to sleep for a specific time. This becomes essential since the implementation of the IEEE 802.11 tech-

nology in the Android devices do not provide full control of the IEEE 802.11 interface, namely to force the wireless interface to sleep.

Finally, the EXPoSE framework also aims to overcome the limitations of the implemented power saving techniques in Android devices, by achieving a good trade-off between the energy consumed and the IEEE 802.11 interface performance in the presence of Continuous Media Applications.

5.2 Architecture and Specification

This section presents the architecture and specification of the EXPoSE framework that was designed to meet the above goals, namely the control of the IEEE 802.11 interface sleep periods, in Android devices, by the end-users and/or applications.

In order to specify and design the EXPoSE architecture, it was important to understand how the IEEE 802.11 technology is implemented in Android devices. Therefore, the Android open source code was studied to ascertain how the IEEE 802.11 components interact and what are the available functionalities to control the network interface. Based on this study, Figure 5.1 was created to depict the IEEE 802.11 architecture and the interaction between the different components.

As shown, the IEEE 802.11 implementation is mostly developed in the user space. However, the entire communication flow ends in the IEEE 802.11 driver implemented in the kernel space.

When an end-user or application intend to manage something related to the wireless interface, must firstly communicate with the Android framework Application Programming Interface (API). This API is provided by the Android Software Development Kit (SDK), based on Java language, which allows to control some of the IEEE 802.11 functionalities (e.g. connect, disconnect, scan, etc). However, this framework does not provide any interface to control the wireless interface power save modes neither the sleep periods.

By performing a request through the Android framework API, the system services will be binded by the "*Wifi Manager*". Then, a message is sent to the "*Wifi Service*" that communicates with the other services related with the IEEE 802.11 technology. The "*Wifi Native*" is responsible for the interaction with the low level layer through the Java Natural Interface (JNI), which implements the native methods, in C or C++ languages, called by the service. Then, the native methods make usage of a socket to send messages to the "*wpa_supplicant*" that finally communicates with the IEEE 802.11 driver.

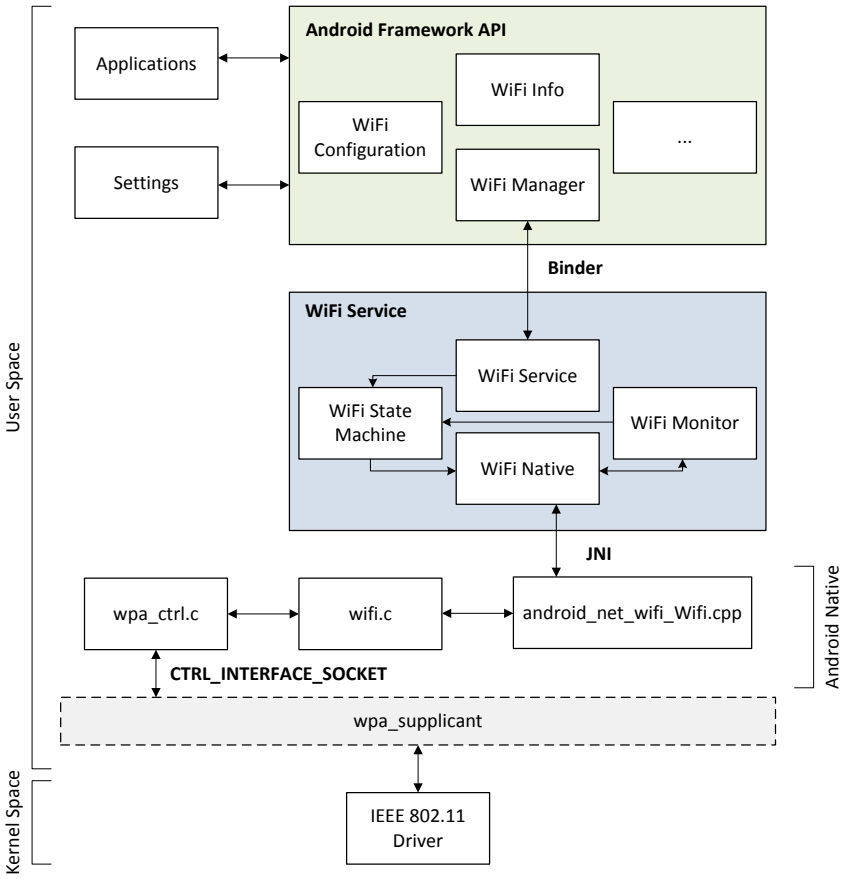


Figure 5.1: IEEE 802.11 simplified architecture in Android. Based on [Amuhong, 2014].

The driver consists in a set of kernel modules which interact between each other in order to control anything related to the wireless interface. A more detailed description of the IEEE 802.11 technology implementation in Android devices and also the presentation of the main layers in the Android OS can be found in Appendix B.

By analyzing the kernel source code, it was possible to verify that at the kernel level it is possible to control the IEEE 802.11 interface power management modes. However, there is no provided interface to force the wireless card to sleep for a given time. Therefore, the EXPoSE framework was created, to give the full control of the wireless interface to the end-users.

Figure 5.2 shows the EXPoSE framework architecture which is composed by two layers: an Android service (EXPoSE Service), present in the user space, and a low level module implemented in the kernel (EXPoSE kernel module).

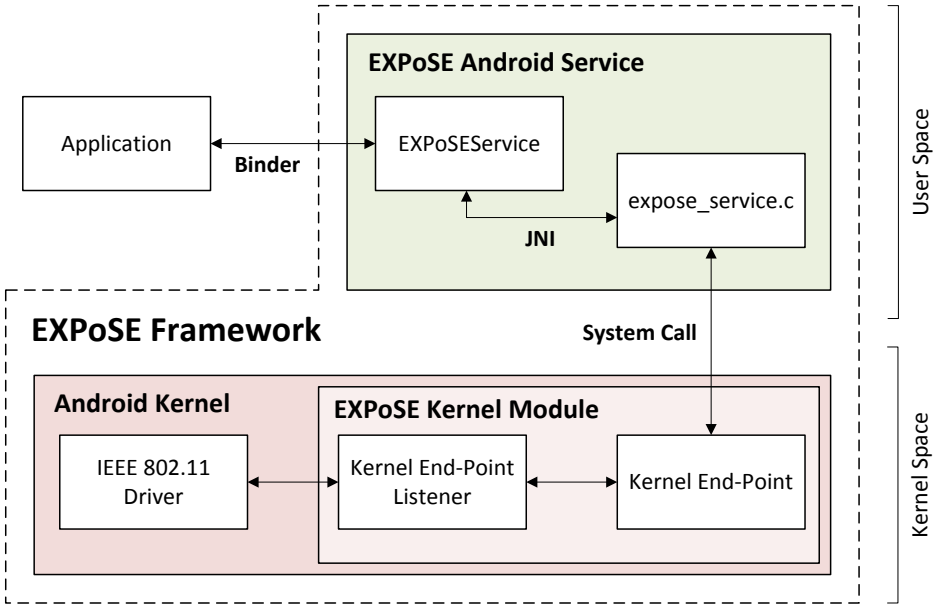


Figure 5.2: EXPoSE framework architecture.

Regarding the communication with the framework, the EXPoSE service allows an end-user and/or application to control the IEEE 802.11 interface by configuring the service in three distinct ways:

- **Power Mode definition:** enables the application to change the IEEE 802.11 mode to No-PSM, Adaptive-PSM or Legacy-PSM;
- **Pattern-based:** enables the application to set the IEEE 802.11 interface in a sleep/awake pattern along time. The application must configure the time that the interface must be awake, α milliseconds, and the time that it must be sleeping, β milliseconds;
- **Maximum Allow Delay (MAD) definition:** allows the application to configure which maximum delay of milliseconds is supported. In this mode, the awake/sleep pattern will be set by the EXPoSE service, taking into account the maximum delay supported by the application.

The service is also responsible for the control of the communication between the user space and the kernel space, through the "*Kernel End-Point*". This extends the IEEE

802.11 implementation to allow the control of the wireless interface from higher layers, rather than in kernel space.

The EXPoSE kernel module was developed since the Android devices do not provide any interface to change the IEEE 802.11 power saving modes at the user space. Therefore, this module contains a "*Kernel End-Point*" which allows the communication between the user and kernel space and, a "*Kernel End-Point Listener*" that aims to perform the communication between the end-point and the driver to change the IEEE 802.11 interface power modes. Furthermore, since the IEEE 802.11 driver does not have an interface to force the wireless card to sleep for a given time, the "*Kernel End-Point Listener*" provides this functionality. This is possible by configuring the Legacy-PSM mode to only awake the wireless interface in the *Beacon* interval after a selected time to sleep.

The communication with the driver is done by the listener in order to achieve two goals: send commands to the driver as soon as the end-point changes and avoid any extra energy costs. This is possible by implementing a lock mechanism using semaphores which will be further detailed, in the next section.

Generally, when the EXPoSE service receives a message from an application, indicating that the IEEE 802.11 interface power mode must be changed, the service will change the "*Kernel End-Point*" that wakes up the "*Kernel End-Point Listener*". Then, the listener sends a command to the wireless driver that finally changes the power save mode.

When the pattern-based approach is configured, the application fills three values: a flag that indicates if the pattern must be repeated over time and the α and β values corresponding to the time awake and the time in sleep, respectively. In this mode the EXPoSE service will stay in loop sleeping for α more β milliseconds after informing the "*Kernel End-Point*" to send the interface to sleep for β milliseconds.

When the MAD approach is used, the application only needs to inform the maximum delay tolerated by the application, γ. The EXPoSE service is responsible to set the α and β values to configure the sleep/awake pattern, taking into account the delay restriction. Then, it acts as the pattern-based approach.

The EXPoSE framework was based on a mechanism proposed by Pyles et al. [Pyles et al., 2011], named SiFi, which aims to control the IEEE 802.11 interface power modes, in Android devices. In order to implement the SiFi mechanism, the "*wpa_supplicant*" was extended to be able to receive a command which informs how long the wireless interface must operate in Legacy-PSM. When this command is received, the "*wpa_supplicant*" forces the IEEE 802.11 driver to use the Legacy-PSM mode and waits for a certain

time after switching back the interface to be constantly awake. To allow applications to send commands to the "*wpa_supplicant*", another component was created, the "*WiFi Manager*". This component is a daemon, implemented outside the Android virtual machine, which is able to communicate with the "*wpa_supplicant*" and is constantly listening to a queue that is also accessed by applications. The applications send messages to the queue in a First In, First Out (FIFO) logic and the "*WiFi Manager*" consumes these messages and informs the "*wpa_supplicant*" to change the wireless interface into Legacy-PSM for a given time. In order to start and stop the "*WiFi Manager*" daemon, the init scripts of the mobile were modified.

Although the mechanism proposed by Pyles et al. was used as an inspiration for the EXPoSE framework, the designed architecture is different, as shown before. Moreover, the EXPoSE framework provides more functionalities, namely the pattern-based and maximum allowed delay approaches which uses the Legacy-PSM mode with a specific configuration to ensure that the wireless interface is always sleeping in the sleep periods. As opposed, the SiFi mechanism only changes the wireless interface power saving modes between No-PSM and Legacy-PSM, implying that when operating in Legacy-PSM, the wireless interface still needs to wake up to receive the *Beacon* frames.

5.3 Implementation

This section discusses the details of the EXPoSE framework implementation that was divided in two phases: the implementation of the EXPoSE service, presented in Subsection 5.3.1, and the kernel module development, discussed in Subsection 5.3.2. The implementation of this framework was performed in the LG p990 already specified in Chapter 4.

5.3.1 EXPoSE Service

The EXPoSE service was developed as an Android bound service which is an application component able to perform background operations (i.e. does not contain an user interface) similar to a daemon process. However, this service does not run permanently in background, it starts when an Android component binds it and stops when all the connected components unbind it. Therefore, the EXPoSE service only runs when it is needed not spending additional energy.

Regarding the communication with the framework, the EXPoSE service implements an interface which allows any Android application or service to bind it in order to send

messages. The messages are received in a FIFO logic and depending on the message, the service will operate in three different ways, defined before: Power mode definition, pattern-based approach or maximum allow delay approach.

When the service receives a message, it needs to communicate with the kernel space through the "*Kernel End-Point*". This communication was implemented using the Android Native Development Kit (NDK). The Android NDK allows the usage of the Java Natural Interface (JNI) to implement the communication in low level languages, such as C. Therefore, the implementation was done by performing system calls to change the "*Kernel End-Point*" leading to a simple and high performance in the communication between the user and the kernel space.

5.3.2 EXPoSE Kernel Module

The study and analysis of the kernel default behavior allowed the proposal of modifications to achieve the EXPoSE framework goals. The modifications were implemented as an extension of the default kernel by adding one component, the "*Kernel End-Point Listener*", and expanding the existent "*Kernel End-Point*".

5.3.2.1 Default Implementation

By studying the kernel source code, it was possible to understand the behavior of the kernel concerning the management of the IEEE 802.11 power saving modes.

The default kernel implementation in use allows the wireless interface to operate in three different power modes: No-PSM, Adaptive-PSM and Legacy-PSM. Figure 5.3 shows the default actions of the kernel, regarding the choice of the IEEE 802.11 interface power modes.

When the display of the device is either switched on or off, the kernel triggers an event. If the display is turned on, the kernel decides based on an external variable which mode the IEEE 802.11 interface must operate, namely Adaptive-PSM or No-PSM. When the display is turned off, it also verifies an external variable and switches the IEEE 802.11 mode. However, the selected modes are the Legacy-PSM or the Adaptive-PSM. The selected modes when the display is turned off are the ones that are able to reduce more energy since in this case the user is not using the device and only background traffic is presented.

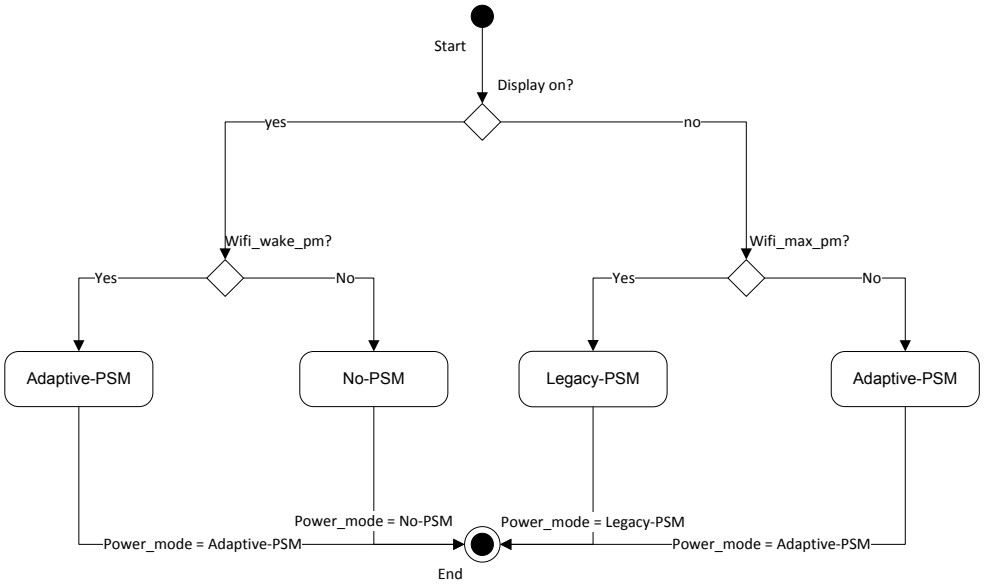


Figure 5.3: Original kernel actions concerning the choice of the IEEE 80211 power management modes.

In addition to the actions performed by the kernel, it is important to understand the interaction between the different components. Figure 5.4 illustrates a possible interaction between the kernel components along the time, concerning the choice of the IEEE 802.11 power management modes when the device display is turned on.

The display is turned on (1), which triggers an event in the kernel. The kernel will verify a variable (2), that is placed in another component, the kernel end-point. After verifying the external variable, the Kernel sends a command to the IEEE 802.11 driver, to set the power management mode. If the variable value is 1, the Adaptive-PSM mode is used (4), otherwise it uses No-PSM (5).

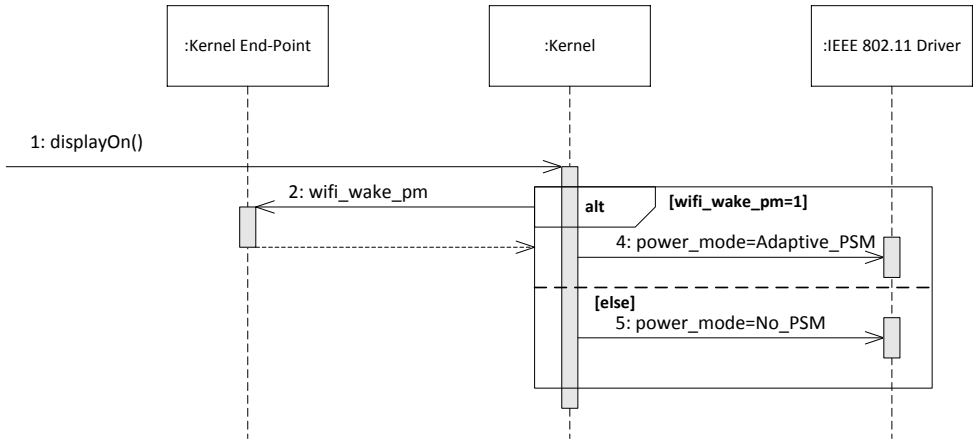


Figure 5.4: Interaction between components of the original kernel related to IEEE 802.11 power management modes, when the display turns on.

Figure 5.5 is similar to the previous, however, it illustrates the interactions when the device display is turned off. When the display is turned off (1), the kernel triggers an event and verifies an external variable stored in the kernel end-point (2). If the variable is set to 1, the Legacy-PSM is used (4). If the value is 0, the kernel informs to use the Adaptive-PSM (5).

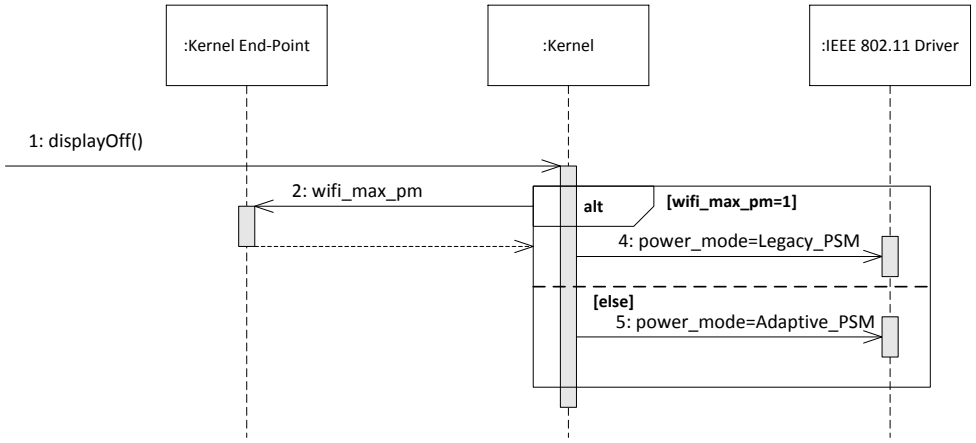


Figure 5.5: Interaction between components of the original kernel related to IEEE 802.11 power management modes, when the display turns off.

Despite that the kernel implementation enables the usage of the three IEEE 802.11 power management modes, there is a gap concerning the change of these modes on the fly. In fact, the kernel only allows the change of a mode when an event is triggered (e.g. the display of the device is switched on or off). Moreover, the default implementation only allows the change of the modes and does not provide the chance of setting the wireless interface to sleep for a given time.

5.3.2.2 Proposed Implementation

In order to overcome the aforementioned problems of the default kernel, the proposed kernel module will extend the default implementation by increasing the options in the "*Kernel End-Point*" and adding a new component, the "*Kernel End-Point Listener*".

The "*Kernel End-Point*" is implemented as an interface (sysfs interface) which contains a set of attributes. These attributes can be changed in the user space and are used by the "*Kernel End-Point Listener*" to understand what action must be performed (e.g. change the wireless interface mode). In order to expand the "*Kernel End-Point*" the attribute "*force_pm*" was created to inform the kernel how long, in milliseconds, the wireless interface must stay in the sleep state.

To address the change of the modes on the fly, the "*Kernel End-Point Listener*" was developed. This listener is a thread that lies on the kernel space and is waiting on a mutex semaphore. This semaphore is released by the "*Kernel End-Point*" every time an attribute is set in the user space. The lock mechanism using the semaphores allows the listener to stay awake only when it is needed. Therefore, it does not incur in any extra energy cost.

Figure 5.6 illustrates a possible interaction between the components of the modified kernel, over the time.

When the display is turned on (1), as in the original version, the kernel triggers an event. However, it will not verify which IEEE 802.11 power management mode will be used. Instead, it simply informs the IEEE 802.11 driver to use No-PSM (2). This is done for the sake of simplicity.

After switching the wireless interface to No-PSM, the kernel starts the "*Kernel End-Point Listener*" (3). As previously stated, the listener waits for a semaphore to be released by the "*Kernel End-Point*" in order to avoid spending extra energy. When an attribute in the end-point is changed, as in (4) and (6), the listener wakes up and sends a command to the driver as showed in (5) and (7). In these particular cases, the changed attributes only inform which power mode must be used by the wireless interface. On

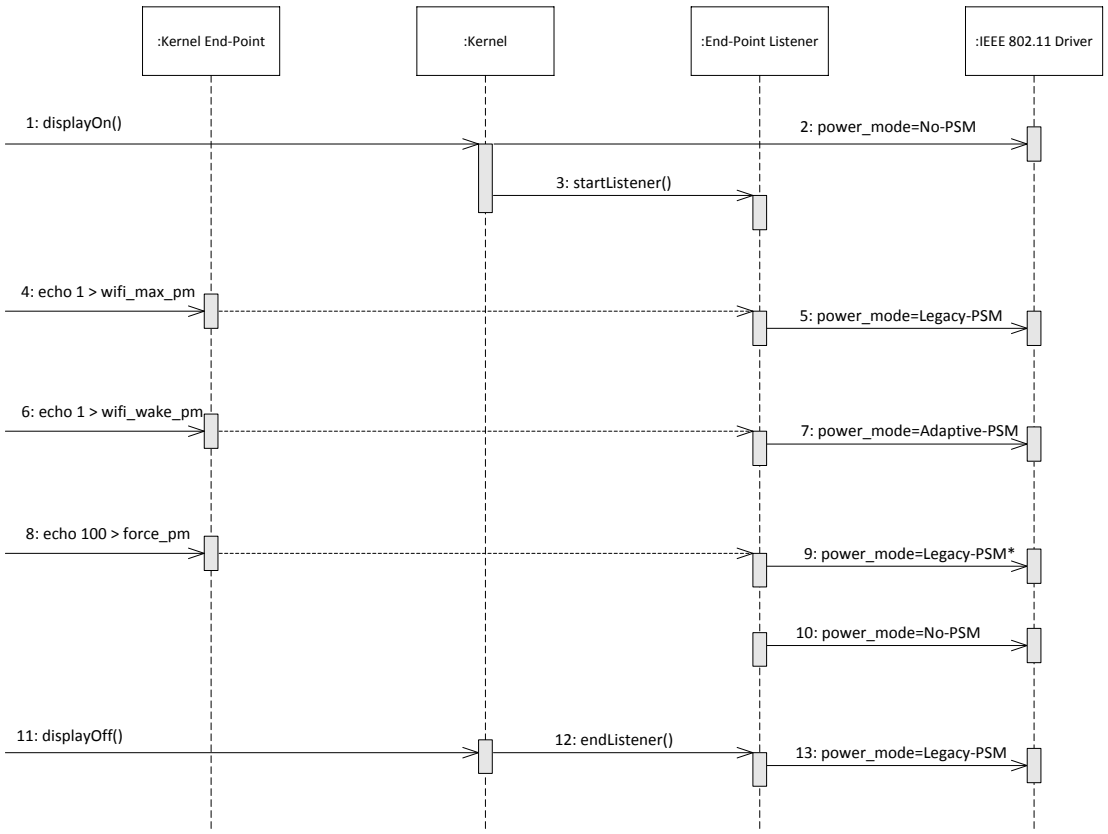


Figure 5.6: Interaction between components of the modified kernel related to IEEE 802.11 power management modes.

the other hand, in the message (8) the "*force_pm*" value is changed in order to force the interface to sleep for 100 ms. Therefore, the listener wakes up and sends a message to the driver to inform that the Legacy-PSM must be used (9). Additionally, the Legacy-PSM configuration parameters are adapted to ensure that the interface will be always sleeping when this mode is employed, forcing the packets to be buffered in the AP during this period. Then, the listener sleeps for the amount of time specified in the end-point (100 ms) before changing the wireless interface to operate in No-PSM mode. This allows the pending data to be transmitted without any extra signaling (e.g. polling messages).

Finally, when the display is turned off (11) the kernel informs the listener that it must stop (12) and send the wireless interface to Legacy-PSM (13).

5.4 Validation

The validation of the EXPoSE framework was performed in the testbed already described in Chapter 4. Therefore, the analyzed results will be compared with the previous discussed tests and are divided as follows: first, the pattern-based sleep approach will be presented, followed by the analysis of the maximum allowed delay approach.

5.4.1 Pattern-Based Approach

The pattern-based approach was evaluated by setting a fixed packet size of 1000 bytes and a fixed constant transmission rate of 200 packets per second. This configuration was chosen since the Legacy-PSM performance in this scenario becomes worst than No-PSM and Adaptive-PSM modes, as shown in Chapter 4.

In order to study the impact of the EXPoSE sleep pattern mechanism in the energy consumption and QoS, 9 distinct sleep/awake patterns were configured.

Table 5.1 summarizes the selected configurations and depicts a relationship between the awake and sleep periods, represented by κ which is calculated through Equation (5.1).

$$AwakePeriod = \kappa \times SleepPeriod \tag{5.1}$$

Table 5.1: EXPoSE pattern-based configurations

Test ID	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**
Loop Flag	1	1	1	1	1	1	1	1	1
SleepPeriod (ms)	30	30	30	60	60	60	120	120	120
AwakePeriod (ms)	90	30	10	120	60	20	360	120	40
κ	3	1	1/3	3	1	1/3	3	1	1/3

As shown, in each test, the awake/sleep pattern will be repeated along the time. The sleep periods selected were 30, 60 and 120 ms and, for each one, it is selected one awake period with a κ equals to 3, 1 and 1/3. These κ values were chosen since they represent an amount of time where the interface will be awake by 75%, 50% and 25% of the total test time (60 seconds).

Figure 5.7 depicts the energy savings (y-axis) introduced by the EXPoSE pattern-based approach compared with the Adaptive-PSM mode.

The results show a clear relationship between the total time in sleep and the energy savings. In fact, by sleeping more, the wireless interface consumes less energy leading to an overall reduction in the energy consumption. Moreover, it is also possible to establish a relationship between the amount of times we change the network interface to sleep and

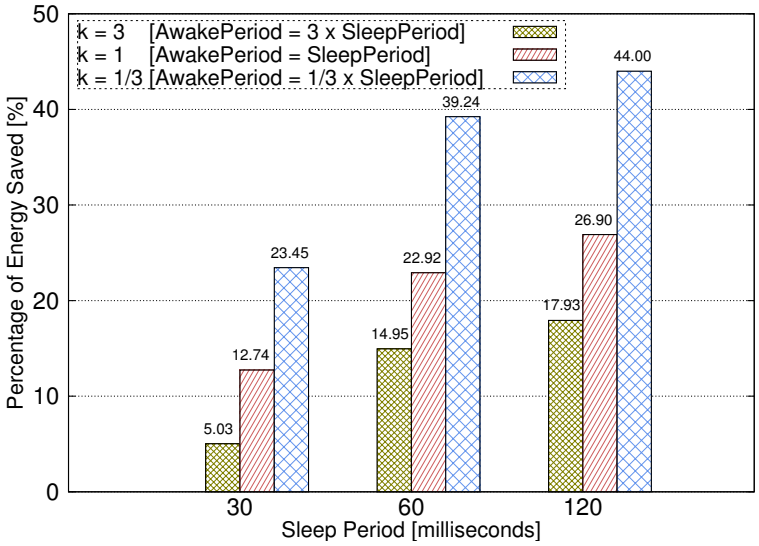


Figure 5.7: Energy savings of the EXPoSE framework pattern-based approach, when compared with Adaptive-PSM.

awake states. Since, for the same amount of time sleeping, the tests when the station changes the interface state less times, consumes lower energy. Therefore, it is possible to understand that, as expected, changing the states has an energy cost associated, as also show by the results in Appendix C.

The results also show that even in the cases where the IEEE 802.11 interface is sleeping only for 25% of the total time, it is possible to save 17.94% of energy when compared with the Adaptive-PSM mode. When the value of κ is set to 1/3 (i.e. the total time in sleep represents 75%) the savings are 23.45%, 39.24% and 44%, respectively. However, despite the energy saving, it is important to understand the impact of the sleep periods in the QoS.

Figure 5.8 shows the one way delay (y-axis) introduced by the EXPoSE pattern-based approach. It is noticeable a linear relationship between the amount of time in sleep and the delay.

As opposed to the No-PSM and Adaptive-PSM modes, the EXPoSE pattern-approach introduces additional delay, since the wireless interface performs a sleep/awake loop along the time.

By analyzing the one way delay results together with the energy savings depicted in Figure 5.7, it is possible to observe a relationship between the delay introduced and the energy saved by the wireless interface. In fact, by increasing the time of the IEEE 802.11 interface in the sleep state, the energy consumption is reduced. However, this reduction

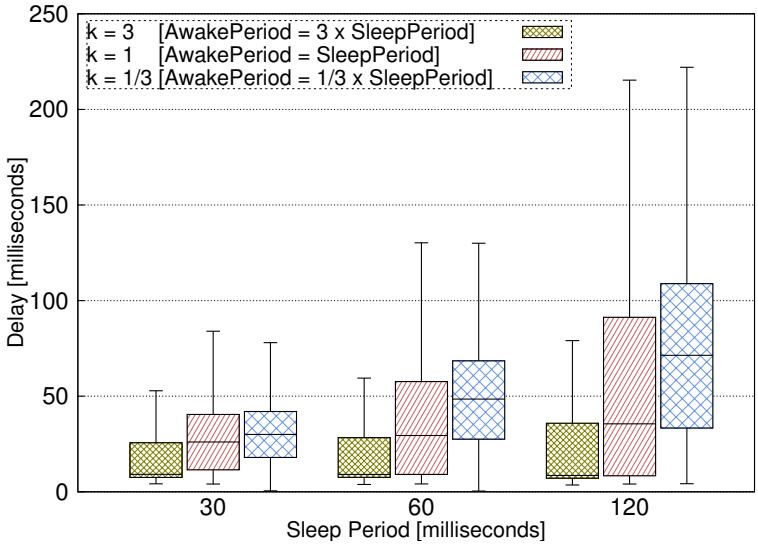


Figure 5.8: One way delay introduced by the EXPoSE framework pattern-based approach.

in the energy has a cost associated, namely the increase of the delay. This happens because when the IEEE 802.11 interface is sleeping the packets are buffered for the station, adding extra delays, but since the interface is in the lowest energy consumption state, the energy is reduced. In fact, the extra delay is needed to reduce the energy consumption. Therefore, the challenge is to find a good trade-off between the extra delay and the energy consumed by the device.

When comparing the Legacy-PSM in the same scenario, the EXPoSE framework shows much better results concerning the introduced delay. The EXPoSE pattern-based approach introduces median delays lower than 80 ms with maximum delays that not reach the 250 ms, as opposed to the Legacy-PSM which introduces a delay with a median around 4 seconds and a maximum reaching the 7 seconds, as showed in Figure 4.9.

To further investigate the QoS performance of the EXPoSE framework, the packet loss will be also analyzed and is presented in Figure 5.9.

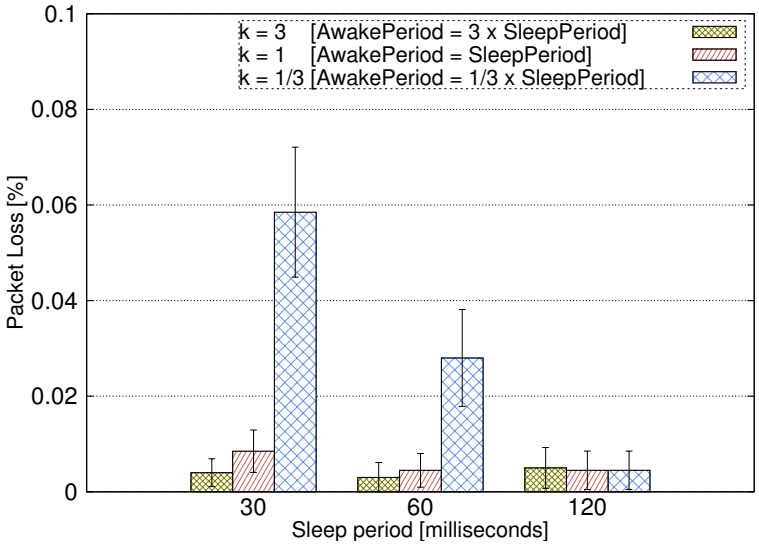


Figure 5.9: Packet loss introduced by the EXPoSE framework pattern-based approach.

The packet loss results show that the EXPoSE sleep pattern approach does not introduce significant losses in the studied scenarios [Y.1541, 2011]. It is noticeable that the packet loss, even being very low, increases when there are more changes between the sleep and awake states. Moreover, when the change between the states is done in higher periods of time the packet loss remains constant. Nonetheless, the packet losses do not exceed the 0.08% which represents a non significant part of the total packets transmitted. Therefore, the introduced packet losses will not affect the energy consumption neither the QoS.

5.4.2 Maximum Allowed Delay Approach

To validate the maximum allowed delay approach it is important to study and explore the configuration of the sleep and awake periods in different scenarios. Therefore, two scenarios were selected: one with a maximum delay tolerated of 100 ms and another with the delay restriction at 200 ms. The maximum allowed delays were selected since they can represent maximum delays acceptable for applications such as VoIP (100 ms) [G.114, 2003] and video streaming (200 ms) [Y.1541, 2011].

As in the validation of the pattern-based approach, discussed in Subsection 5.4.1, the packet size was fixed at 1000 bytes and the transmission rate at 200 packets per second.

To investigate which is the maximum time that the interface needs to be awake in order to maintain the maximum allowed delay restriction (assuming that the interface

will sleep for the maximum delay tolerated) the tests were configured to range the awake period between 50 and 500. Figure 5.10 depicts the percentage of the energy saved (y-axis) for each performed test.

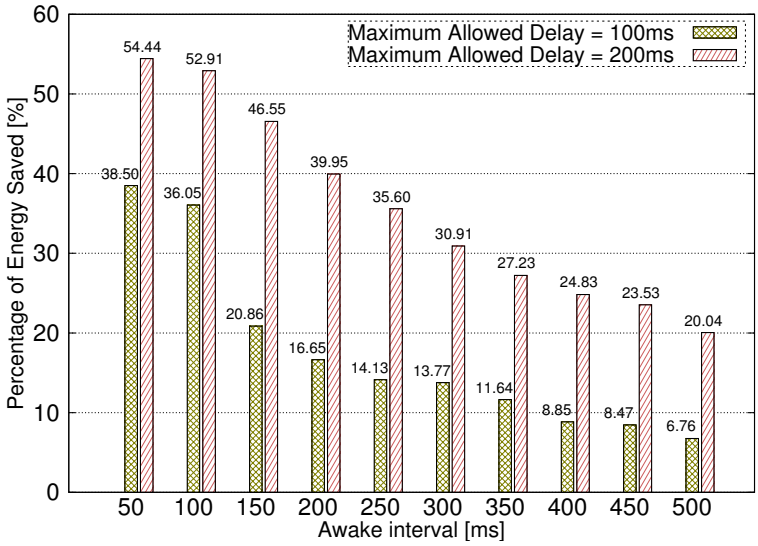


Figure 5.10: Energy savings of the EXPoSE framework maximum allowed delay approach, when compared with Adaptive-PSM.

The results show energy savings that can reach 38.5% and 54.44% for the maximum allowed delays of 100 and 200 ms, respectively. However, it is not guaranteed that this patterns keep the end-user expectations. Therefore, the results of the delay are shown in Figure 5.11.

As already stated before, the results show that the delay is proportional to the device energy consumption. Moreover, the delay is higher when the total time that the interface is in sleep is higher, leading to a decrease of the delay, when the awake interval increases.

The results also show that to maintain the delay bound restrictions, the interface must be awake for 300 ms and 450 ms, for the maximum delay of 100 ms and 200 ms, respectively. This leads to energy savings of 13.77% and 23.53% if the application supports maximum delays of 100 ms and 200 ms. Additionally, the tests when the percentage of the energy saved is higher, showed that the application guaranties the delay restriction for 75% of the packets (third quartile of the boxplot).

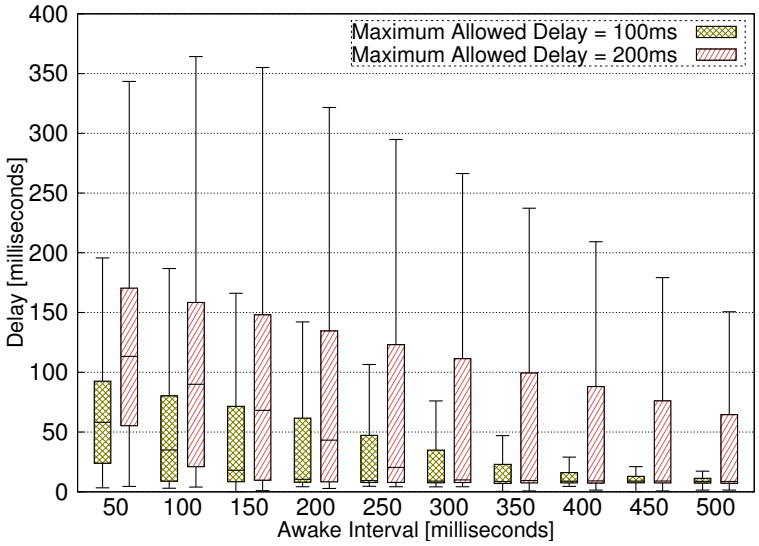


Figure 5.11: One way delay introduced by the EXPoSE framework maximum allowed delay approach.

Furthermore, the packet loss of the maximum allowed delay approach results were also analyzed and are present in Figure 5.12.

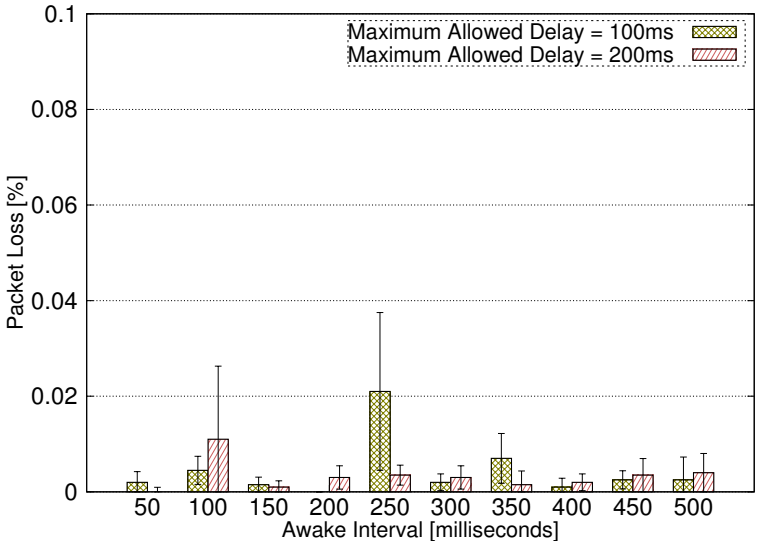


Figure 5.12: Packet loss introduced by the EXPoSE framework maximum allowed delay approach.

The depicted values showed that the EXPoSE framework adds packet losses always below 0.04%. These are negligible values since they represent maximum losses lower than

5 packets in 12000 packets transmitted [Y.1541, 2011]. Therefore, the losses presented do not have an impact in the energy consumption or in the QoS. Moreover, since the loss values are extremely low, a slight variation in only a test, which usually occurs in a real environment, represents an higher variation in the depicted values, leading the values to not represent a trend (e.g. the packet loss remain constant or decrease when the awake interval is higher).

5.5 Summary

This chapter presented the proposed Android framework for Extending Power Saving control to End-users (EXPoSE). The development of this framework becomes essential since it provides a tool that overcomes the existent gaps in the IEEE 802.11 implementation in the Android devices, namely by creating an interface that allows the control of the wireless interface power modes in higher layers (e.g. applications layer) and provides a way to set the interface to sleep for a given time.

As the framework allows the control of the wireless interface, it becomes easy to study and propose new mechanisms concerning energy savings. Furthermore, by giving the control of the framework to the end-users and/or applications, it is possible to keep the desired expectations, namely the maximum delay tolerated.

Using the basic functionalities of the framework, such as, the control of the IEEE 802.11 interface sleep periods, two approaches to reduce the energy consumption in the presence of Continuous Media applications were developed. This implementation leaded to a publication of a paper titled *Towards End-User Driven Power Saving Control in Android devices* [Bernardo et al., 2014], included in Appendix A, and the validation results showed a clear benefit of using the proposed framework.

When comparing the results of the framework, in the tested scenarios, with the current power modes implemented in Android devices, the EXPoSE approaches achieved the lower energy consumption. In fact, the pattern based approach allowed energy savings up to 44% when compared with the Adaptive-PSM mode, while the maximum delay introduced does not reach the 240 ms. On the other hand, the maximum allowed delay approach allows energy savings of 13.77% and 23.53%, compared with the Adaptive-PSM mode, respectively for maximum delays tolerated of 100 and 200 ms. However, if the applications only needs 75% of the packets to be delivered below the delay bound restrictions, the energy savings can reach the 38.50% an 54.44% for the maximum allowed delays of 100 and 200 ms, respectively.

Furthermore, the results showed percentages of packet losses that do not affect the

QoS. Therefore, the EXPoSE framework approaches allowed, in the studied scenarios, the achievement of a good trade-off between the energy consumption and the QoS in the presence of continuous traffic, while giving the control to the end-users, in order to guarantee their expectations.

Chapter 6

OPAMA for Android Devices

This chapter presents the Power save Algorithm for continuous Media Applications (OPAMA). Section 6.1 describes the original version of OPAMA proposed by Bernardo et al. [Bernardo et al., 2013]. Based on the original OPAMA, two versions of this mechanism were developed in this thesis, for Android devices: the OPAMA lite and the enhanced OPAMA lite.

The OPAMA lite, presented in Section 6.2, is a simplified version of the original OPAMA which represents a solution where only the AP needs to be modified. Therefore, the station remains operating in Legacy-PSM mode, but the AP will manage the delivery of the packets differently, as explained next.

The enhanced OPAMA lite, proposed in Section 6.3, is an improvement to the OPAMA lite that avoids the polling phase caused by the Legacy-PSM mode. This phase represents a bottleneck of performance as showed by the results in Chapter 4. Therefore, avoiding this phase will increase the wireless interface performance, concerning the energy consumption and the QoS. As opposed to the OPAMA lite, this mechanism is a solution where both AP and station need to be modified.

6.1 Original OPAMA

This section presents the original Power save Algorithm for continuous Media Applications (OPAMA) proposed by Benardo et al. The mechanism design and specification is discussed first, followed by a description of its core algorithm.

6.1.1 Design

The original OPAMA architecture was designed to meet two main goals: allow the saving of energy by taking into account the end-user expectations and address the existent gap of the current power saving mechanisms, which fail to reduce the devices energy consumption in the presence of Continuous Media Applications.

Figure 6.1 shows an example of OPAMA in action. When the Access Point (AP) receives packets to a given station operating in Power Save Mode (PSM), the packets are buffered in a queue. At each *Beacon* interval, the station wakes up to receive a *Beacon* frame that informs if there is any pending data. As opposed to the Legacy-PSM behavior, where the AP always informs the station of pending data, the OPAMA will make this decision by taking into account the end-user expectations (e.g. the maximum allowed delay).

In Figure 6.1, when the first *Beacon* is sent, the algorithm decides to inform the station of pending data. Therefore, the station sends a *PS-Poll* frame and receives the buffered frames. These frames are sent aggregated by using the A-MSDU. Unlike in Legacy-PSM, the station does not need to poll every buffered packet. Instead, the station only sends a *PS-Poll* frame which informs the AP that it will stay awake to receive the pending packets. Additionally, the *PS-Poll* frame was also extended to contain the station maximum allowed delay that could be dynamically adapted along the time.

After receiving all the packets, the station is able to sleep again until the next *Beacon*. As in Legacy-PSM, the station goes to sleep after verifying that the *MORE* data flag, in the *Beacon* frame, is set to 0.

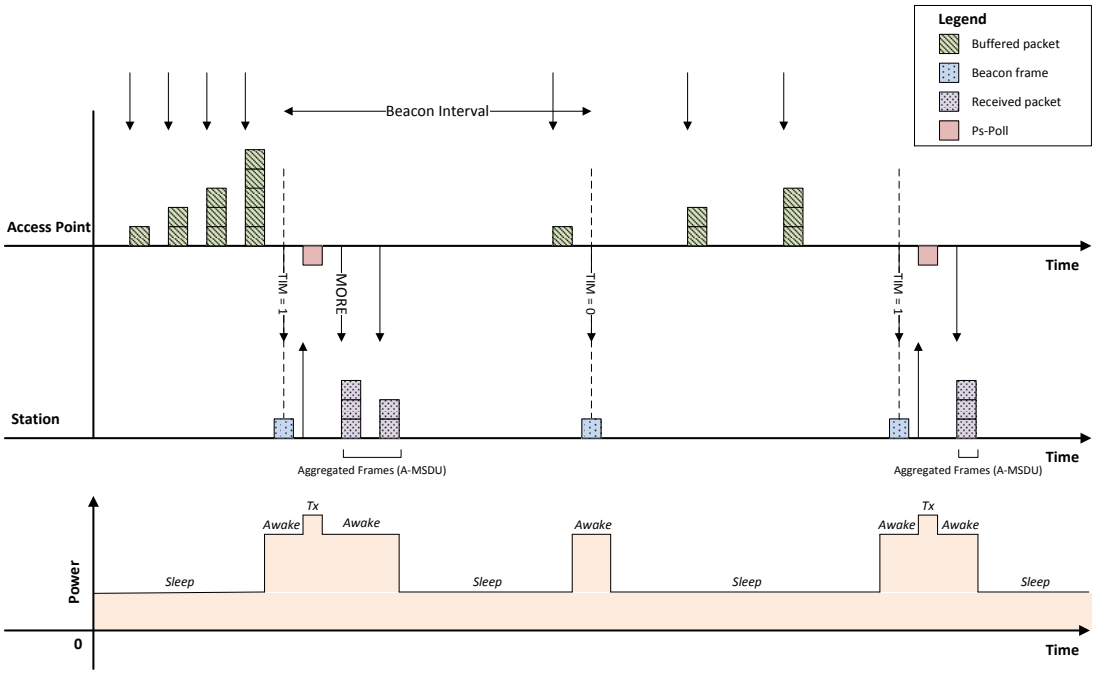


Figure 6.1: Simplified operation of OPAMA and respective energy cost.

When the station wakes up to receive the second *Beacon*, the OPAMA algorithm, even though the AP has buffered packets, hides this information from the station. Therefore, the station will go back to sleep. The AP continues to buffer the sending packets and, at the next *Beacon*, it informs the station that it must stay awake to receive the pending data. As in the first Beacon, the station sends a *PS-Poll* frame and receives the data aggregated, going to sleep after checking that the *MORE* data flag is set to 0.

6.1.2 Original OPAMA Core Algorithm

The OPAMA core algorithm is responsible to decide if the pending data to a given station must be sent or hidden. Algorithm 1 defines the implemented procedure of the OPAMA core algorithm.

First, the algorithm will get the needed variables like the "*maximum allowed delay*", the "*aggregation threshold*" and the "*time until next Beacon*" (*lines 3 to 5*). Then, it iterates over a list of the pending frames. For each frame, the "*current delay*" is checked (*line 11*) and if it reaches the "*maximum allowed delay*" the packets are sent immediately (*lines 13 to 15*). Moreover, the packets are also sent if the "*current delay*"

plus the "*time until next Beacon*" is higher than the "*maximum allowed delay*" (*lines 17 to 19*).

At this point, the OPAMA algorithm is able to maintain the end-user expectations, by controlling the delivery of the packets, taking into account the delay bound restrictions.

Furthermore, the algorithm also includes a specific control mechanism only for video applications. This mechanism is able to control, for example, a streaming quality, by ensuring that a particular type of frame is not buffered over a threshold (*lines 21 to 26*). Additionally, it also enforces a limit on the number of frames that are aggregated (*lines 30 to 32*).

If none of the previous conditions is met, the OPAMA hides the pending frames from the station. Therefore, the station goes to sleep, saving energy.

Despite the fact that OPAMA overcomes the gap of the current power saving mechanisms by reducing energy in the presence of Continuous Media Applications, taking into account the end-user feedback, the authors decided to conduct the OPAMA validation in a simulator. Therefore, the implementation and validation of this mechanism in a real hardware platform, such as the Android OS, contributes towards the green wireless communications, since the Android smartphones are nowadays one of the most widely used mobile devices. Moreover, the validation of OPAMA in real devices will allow to understand the impact of this mechanism in a real environment.

Algorithm 1 OPAMA algorithm based on [Bernardo et al., 2013]

```
1: function OPAMA(MACaddr)
2:     ▷ Get the needed variables
3:     mad ← getMaximumAllowDelay(MACaddr)
4:     aggregationThreshold ← getAggregationThreshold(MACaddr)
5:     timeNextBeacon ← getTimeUntilNextBeacon()
6:     ▷ Iterate over every pending frame
7:     pendingFrames ← getPendingFrames(MACaddr)
8:     pendingBytes ← 0
9:     for each frame in pendingFrames do
10:        ▷ Get the actual frame delay
11:        delay ← getActualDelay(frame)
12:        ▷ If the frame delay is higher than maximum allow delay, send the frames
13:        if delay > mad then
14:            sendBufferedPackets()
15:            return TRUE
16:        ▷ If the sum of the frame delay and the time until the next Beacon is higher
   than maximum allow delay, send the frames
17:        if (delay + timeNextBeacon) > mad then
18:            sendBufferedPackets()
19:            return TRUE
20:        ▷ Specific for video frames:
21:        if getMediaType(frame) == "video" then
22:            ▷ Prevents the "I" frames to be buffered over a threshold
23:            if getFrameType(frame) == "I" then
24:                if getTotalPendingVideoKeyFrames(MACaddr) > α then
25:                    sendBufferedPackets()
26:                    return TRUE
27:        ▷ Increment the total pending frame bytes
28:        pendingBytes ← pendingBytes + getSize(frame)
29:    ▷ Prevents the frames to be aggregated over a threshold
30:    if (pendingBytes/aggregationThreshold) ≥ β then
31:        sendBufferedPackets()
32:        return TRUE
33:    ▷ Hide the buffered packets to the station
34:    hideBufferedPackets()
35:    return FALSE
```

6.2 OPAMA Lite

This section presents the lite version of OPAMA which was implemented and evaluated in a real testbed. First, the design of the OPAMA lite is discussed. Then, the core algorithm is described and compared with the original. The implementation of this mechanism is presented next and, finally, the OPAMA lite validation in testbed is analyzed.

6.2.1 Design

The OPAMA lite was developed based on the original OPAMA. However, some simplifications have been done. In OPAMA lite only the AP is modified, avoiding thus changes in the mobile device. Moreover, OPAMA lite does not perform aggregation.

Figure 6.2 depicts an example of the OPAMA lite behavior. As in the original version, when the AP receives packets to a station that is operating in Legacy-PSM, the packets are buffered in queues. The station wakes up to receive the *Beacon* frames at each *Beacon* interval and the core algorithm will decide if the pending data should be sent or hidden.

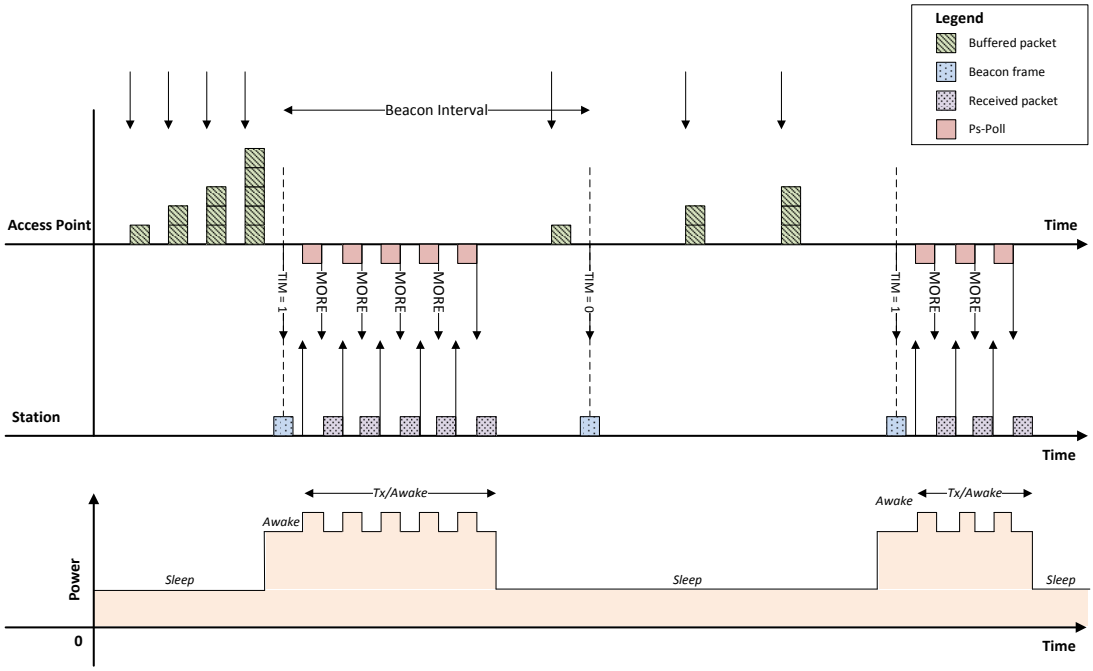


Figure 6.2: Simplified operations of OPAMA lite and respective energy cost.

When the AP sends the first *Beacon*, the frame informs the station that it must stay awake until it has received all the buffered packets. The station responds sending a *PS-Poll* frame and receives one pending frame as response. Since the implementation of the OPAMA lite is only performed at the AP level, the station is operating in the default Legacy-PSM mode. Therefore, it needs to poll every pending packet before going to sleep.

In the second *Beacon*, the AP decides to hide the queued packets by setting the Traffic Indication Map (TIM) bit to 0, allowing the station to sleep until the next *Beacon*. The last *Beacon* frame informs that the AP has buffered packets to the station and, therefore, the station polls every pending packet until the *MORE* data field indicates that there are no more pending packets. As a result, the station is able to sleep again.

When comparing the power plotted in Figure 6.2 to the one in Figure 6.1, it is evident that the OPAMA lite is more energy expensive than the original OPAMA. This happens due to the polling phase which occurs since the station is operating in the default Legacy-PSM mode. Moreover, the aggregation of the packets is also not contemplated, leading the station to poll every pending packet.

In fact, the OPAMA lite implementation has a similar behavior when compared with the Legacy-PSM mode. However, unlike in Legacy-PSM, the OPAMA lite allows the control of the time that the packets stay in the buffered queues, extending the Legacy-PSM default behavior to be controlled by the end-user expectations, namely the maximum allowed delay.

6.2.2 OPAMA Lite Core Algorithm

Algorithm 2 presents a simplification of the OPAMA core algorithm that decides if the pending data must be sent or hidden.

Before the AP sends the *Beacon* frames, the OPAMA lite algorithm runs and verifies the station "*maximum allowed delay*" and the "*time until next Beacon*" (*lines 3 and 4*). Then, for each pending packet, the total time of the packet in the queue is calculated. If this time is higher than the "*maximum allowed delay*" or if this time plus the "*time until next Beacon*" is higher than the "*maximum allowed delay*", the pending data is sent. As opposed to the current delay calculated by the original OPAMA, the OPAMA lite only gets the total time of the packet in the AP queue, for the sake of simplicity. Since the machine that sends the packets is physically connected by an Ethernet cable to the AP, the time of the packet in the queue will be roughly the same as the time difference between the moment when the packet is sent and the actual time. Therefore,

Algorithm 2 OPAMA lite algorithm

```
1: function OPAMA(MACaddr)
2:     ▷ Get the needed variables
3:     mad ← getMaximumAllowDelay(MACaddr)
4:     timeNextBeacon ← getTimeUntilNextBeacon()
5:     ▷ Iterate over every pending frame
6:     pendingFrames ← getPendingFrames(MACaddr)
7:     for each frame in pendingFrames do
8:         ▷ Get the amount of time that the frame has been in queue
9:         timeInQueue ← getTimeInQueue(frame)
10:        ▷ If the time of the frame in queue is higher than maximum allow delay, send the frames
11:        if timeInQueue > mad then
12:            sendBufferedPackets()
13:            return TRUE
14:        ▷ If the sum of the frame time in queue and the time until the next Beacon is higher than maximum allow delay, send the frames
15:        if (timeInQueue + timeNextBeacon) > mad then
16:            sendBufferedPackets()
17:            return TRUE
18:    ▷ Hide the buffered packets to the station
19:    hideBufferedPackets()
20:    return FALSE
```

this simplification will not compromise the OPAMA validation. Furthermore, in a real environment, if the machines are not synchronized the current delay can not be calculated properly.

If none of the pending packets satisfies the above conditions, the packets can stay in the queues and, therefore, the station can go back to sleep.

6.2.3 Implementation

The OPAMA lite implementation was performed fully in the *mac80211* [MAC80211, 2014], a kernel module which implements the IEEE 802.11 stack in the Linux kernel. The choice to implement the OPAMA lite in the *mac80211* was taken since, in this module, it is possible to perform any needed modification of OPAMA at the AP level, such as the control of the queues to buffer the packets and the change of the *Beacon* frames. Moreover, as the *mac80211* driver is part of the Linux kernel tree, it is open source. Therefore, as opposed to other AP implementations, it is possible to study and modify the kernel source code.

The OPAMA lite core algorithm was implemented to run before the *mac80211* module sends a *Beacon* frame. This allows the change of the TIM present in the *Beacon* frame, to hide packets from a given station. In order to change the TIM information, the function that forces the recalculation of the TIM was modified. This function was extended to receive one parameter which informs if the packets must be hidden. If the parameter is *TRUE*, the value of the TIM for that station is set to 0, otherwise, it acts as normal.

When the OPAMA lite core algorithm runs, it needs to iterate over every pending queue and, for each packet, verifies the amount of time in the queue. If this time or this time plus the time until the next *Beacon* are higher than the maximum delay tolerated by the station, the AP informs the station of pending packets. As opposed to the original OPAMA, the "*maximum allowed delay*" can not be set dynamically by sending this information in the PS-Poll frames, since this would require modifications in the mobile device. Instead, the AP makes usage of a configuration file which contains the MAC address of the station and the correspondent "*maximum allowed delay*". The structure that contains the information about the station was extended to contain the "*maximum allowed delay*" value and is used as a fixed parameter.

6.2.4 Validation

The validation of the OPAMA lite mechanism was conducted in the same testbed described in Chapter 4 and under the same conditions. However, due to the need to perform modifications in the AP, instead of using a commercial AP, an Asus Eee PC was used, running Debian 7.0 with the "*3.2.0-4-686-pae*" kernel version. The netbook also runs the hostapd program [Hostapd, 2014] which is a daemon that allows it to act as an AP and to control authentication. In addition to the different AP, its configuration is also different, as shown by Table 6.1.

Table 6.1: AP main configurations

Parameter	**Value**
Beacon interval	100 ms
DTIM interval	1
Buffer size	64 packets

Despite the fact that the *Beacon* interval remains the same, the *DTIM* interval is configured to 1 instead of 3. This configuration was chosen since the original OPAMA was designed with the assumption that the AP sends a TIM in each *Beacon* to inform if there is pending data for a station. Concerning the buffer, its size was configured to queue at most 64 packets, the default value in the kernel, as opposed to the commercial

AP which has a size of 1024 packets.

Since the AP and the configuration used in this testbed are different from the previous results presented in Chapter 4, the validation of the OPAMA lite was performed in two steps: first, the IEEE 802.11 power management modes were characterized in the presence of continuous traffic, representing the baseline results, and then, the OPAMA lite performance was compared with these results.

As in the tests presented in Chapter 4 concerning the Continuous Media Applications, the performed tests were conducted in a scenario where the Android device is receiving continuous UDP traffic duration 60 seconds. All the results are based on 20 runs and are exhibited in the charts with a confidence interval of 95% assuming that the measurements follow a normal distribution.

6.2.4.1 Baseline Results

The baseline results only contemplate the impact of the transmission rate in the No-PSM, Adaptive-PSM and Legacy-PSM modes, since, as showed in Chapter 4, the packet size has no impact in the energy consumption neither in the QoS of these modes.

Figure 6.3 depicts the total energy consumption in Joule (y-axis) by the IEEE 802.11 interface, when employing the three different power management modes in the presence of continuous UDP traffic with different transmission rates (x-axis). The packet size in these tests was set to 1000 bytes and the transmission rate was changed between 50 and 250 packets per second, with steps of 50 packets per second. These test configurations were chosen by the same reasons presented in Chapter 4.

The results show, as in Chapter 4, that the energy consumed by the wireless interface is proportional to the transmission rate. Furthermore, the energy consumed while operating in No-PSM and Adaptive-PSM modes is identical, since, in both modes, the wireless interface needs to stay always awake due to the continuous traffic. However, as opposed to the previous results, where the Legacy-PSM only reduced energy for lower rates, these results show that the Legacy-PSM mode consumes the lowest energy in all the studied transmission rates. To further investigate this energy difference, one way delay and packet loss were also analyzed.

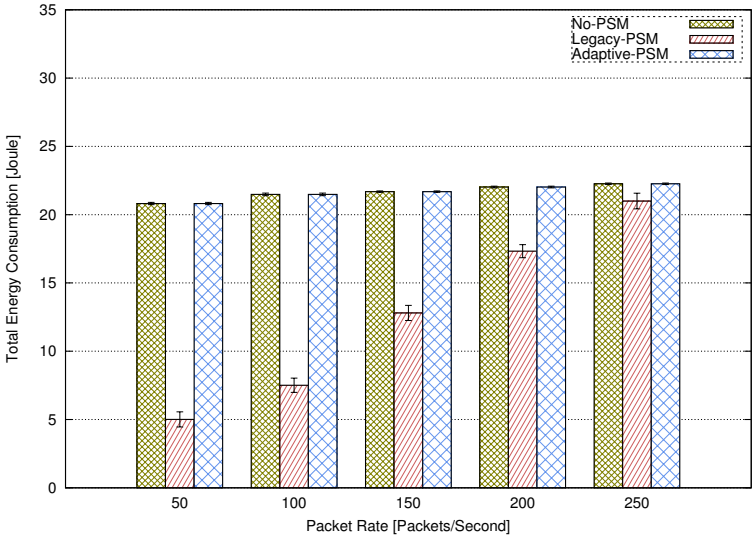


Figure 6.3: Total energy consumed by the IEEE 802.11 interface using three different power management modes, with different transmission rates.

Figure 6.4 shows the one way delay introduced by the Legacy-PSM mode (y-axis) when receiving traffic with different transmission rates (x-axis).

The obtained results show median delays around 50 ms and maximums that do not reach 160 ms. These values represent delays much lower than the obtained in Chapter 4, where the median delays are always higher than 100 ms and, in the highest rates, the median delays exceed the 5 seconds.

These results can be explained since the AP configurations are different. While the *DTIM* interval in this AP is set to 1, forcing the station to wake up at each *Beacon* interval and remain awake to poll every queued packet, in the previous results the *DTIM* interval was 3. Therefore, the station only awakes at each third *Beacon* (300 ms) causing the AP to buffer much more packets that need to be polled when the station is awake, increasing the delay.

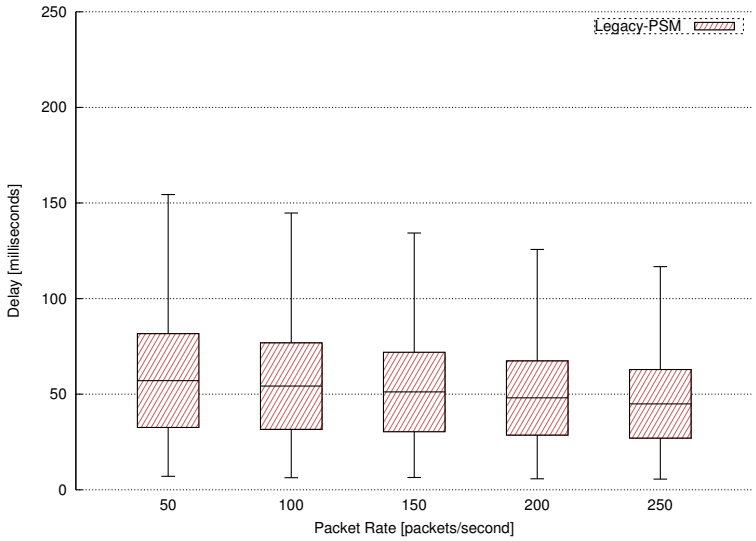


Figure 6.4: One way delay introduced by Legacy-PSM one way with different transmission rates.

Despite that the lower delays can be explained by the AP configurations, this does not explain the tendency of the delay to become lower when the transmission rate is increased. In order to understand this unexpected behavior, Figure 6.5 depicts the percentage of packet loss (y-axis) for each transmission rate (x-axis) when the wireless interface is operating in Legacy-PSM mode.

The results show a direct relationship between the percentage of packet loss and the packet rate. When the packet rate is increased, the packet loss also increases which explains the trend of the delay, since, the oldest packets in the queues are removed due to lack of space. The size of the queues, that allows only to buffer 64 packets, also explains why the results in Chapter 4, where the queue is able to buffer 1024 packets, presented much higher delays.

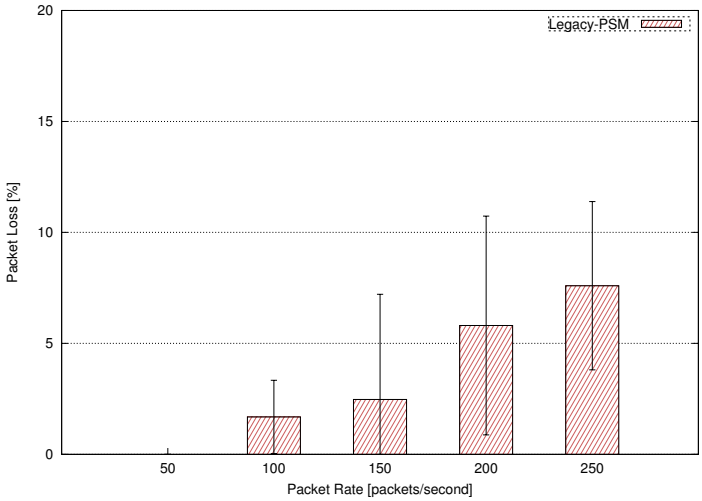


Figure 6.5: Packet loss caused by Legacy-PSM with different transmission rates.

Finally, Figure 6.6 compares the one way delay (y-axis) when the IEEE 802.11 interface is using the No-PSM and Adaptive-PSM modes.

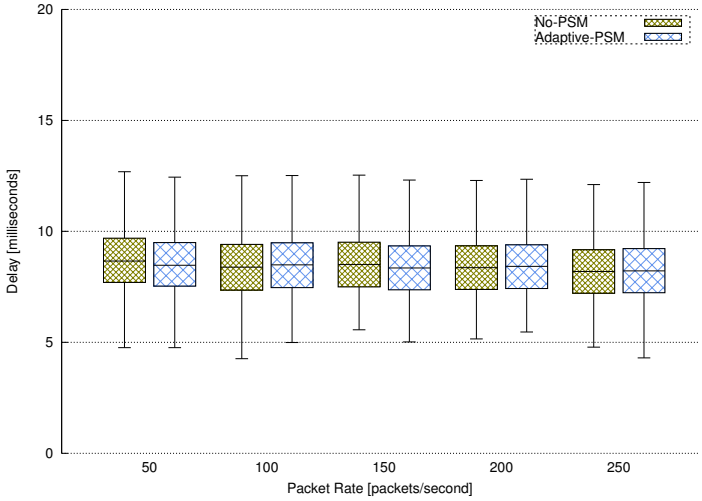


Figure 6.6: One way delay introduced by No-PSM and Adaptive-PSM with different transmission rates.

Like in Chapter 4, the depicted results show median delays varying between 7 and 9 ms, while the maximum delay never reach the 14 ms. Moreover, it is noticeable that the delay remains constant when the transmission rate is increased, allowing to conclude that the transmission rate does not affect the No-PSM and Adaptive-PSM modes delay.

6.2.4.2 OPAMA Lite Validation

As explained before, the lite version of OPAMA is similar to the Legacy-PSM mode, which is used by the station in the OPAMA lite mechanism. However, as opposed to the standard specification, where an AP needs to inform the stations of pending data in each *Beacon*, the OPAMA lite informs or hides this information, depending on the station maximum allowed delay. Therefore, if the station maximum allowed delay is set to 100 ms, the information of pending packets will be presented in each *Beacon* frame and the OPAMA lite behavior must be similar to the Legacy-PSM. To validate this, the maximum delay allowed by the station was configured to 100 ms and the obtained results were compared with the Legacy-PSM mode.

Figure 6.7 depicts the total energy consumption by the wireless interface (y-axis) when using the Legacy-PSM mode and the OPAMA lite mechanism. For each test, the packet size was set to 1000 bytes and the packet rate changed between 50 and 250 packets per second, as in the baseline results.

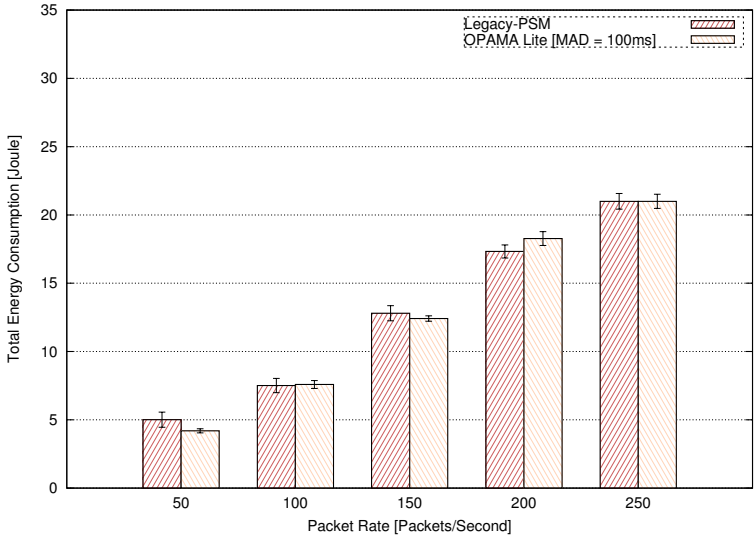


Figure 6.7: Comparison between the total energy consumed by the IEEE 802.11 interface using the OPAMA lite mechanism and the Legacy-PSM mode with different transmission rates.

Despite of the normal slight variations in the energy, when comparing the plotted results it is possible to conclude that when the OPAMA lite mechanism is used, the energy consumption is identical to when the Legacy-PSM mode is employed. Therefore, concerning the energy consumption by the IEEE 802.11 interface, the presented results show that the OPAMA lite mechanism does not add extra energy costs and does not

change the energy performance achieved by the Legacy-PSM mode.

In order to investigate if the OPAMA lite implementation has impact in the Legacy-PSM performance, the QoS metrics, namely one way delay and packet loss, were also analyzed.

Figure 6.8 compares the one way delay (y-axis) introduced by the OPAMA lite mechanism and the one way delay in the Legacy-PSM mode.

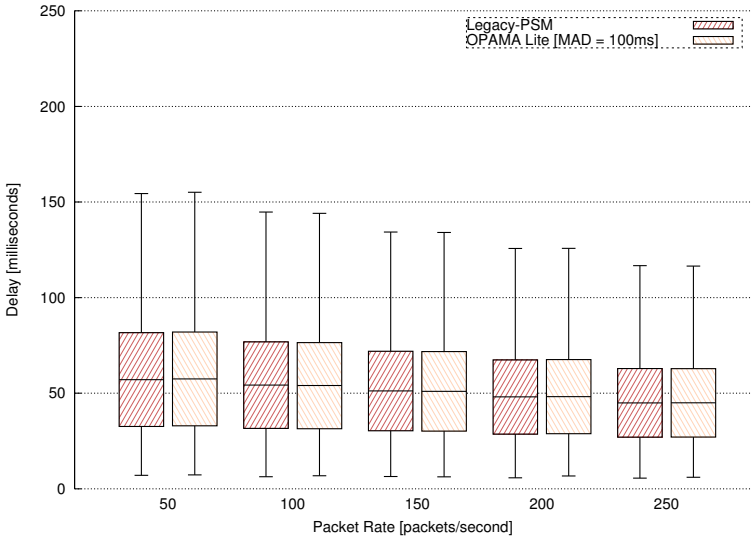


Figure 6.8: Comparison between the delay introduced by the OPAMA lite mechanism and the Legacy-PSM mode with different transmission rates.

By analyzing the depicted results, there is no visible differences between the delay in the OPAMA lite mechanism and the delay in the Legacy-PSM mode. This reinforces the idea that the OPAMA lite has a similar behavior when compared with the Legacy-PSM mode, since, by configuring the station maximum allowed delay to 100 ms, the OPAMA lite core algorithm will inform in each *Beacon* frame, if there is pending data. However, these results also show that the maximum allowed delay of 100 ms is not guaranteed, since, even informing the station of pending data in each *Beacon*, the station needs to poll every single packet, delaying the packets delivery.

Figure 6.9 presents the percentage of packet loss (y-axis) when the OPAMA lite mechanism and the Legacy-PSM mode are used with the different transmission rates.

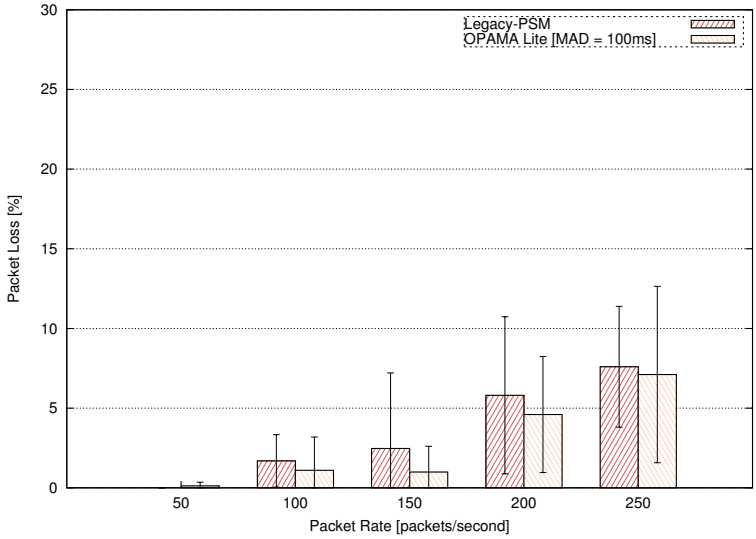


Figure 6.9: Comparison between the packet loss introduced by the OPAMA lite mechanism and the Legacy-PSM mode with different transmission rates.

Despite of the variations in the percentage of packet loss which, as seen by the error bars, are normal, both approaches have similar packet losses. Therefore, it is possible to conclude that the OPAMA lite mechanism does not change the Legacy-PSM performance regarding the packet loss.

The results with the maximum allowed delay configured at 100 ms validated that the OPAMA lite mechanism implementation does not differ the Legacy-PSM performance. However, to fully study and validate the OPAMA lite, the management of the buffering queues was also assessed. Furthermore, the OPAMA lite core algorithm, which decides if the station must be informed of pending data, was also validated.

In order to validate the OPAMA lite core algorithm and the management of the buffering queues, the maximum allowed delay by a station was changed between 100 and 400 ms, with steps of 50 ms. Since the Legacy-PSM mode already presents packet losses when the transmission rate is 100 packets per second, the next performed tests were conducted with a transmission rate of 50 packets per second.

Figure 6.10 depicts the total energy consumed by the wireless interface (y-axis) when the OPAMA lite is employed and the station allows different maximum delays (x-axis).

The depicted results show only slight variations in the energy consumption of the IEEE 802.11 interface when the maximum allowed delay is increased. In fact, the energy consumption is almost constant. These results can be explained due to the polling phase, since, by increasing the maximum allowed delay the time that the packets remain in the

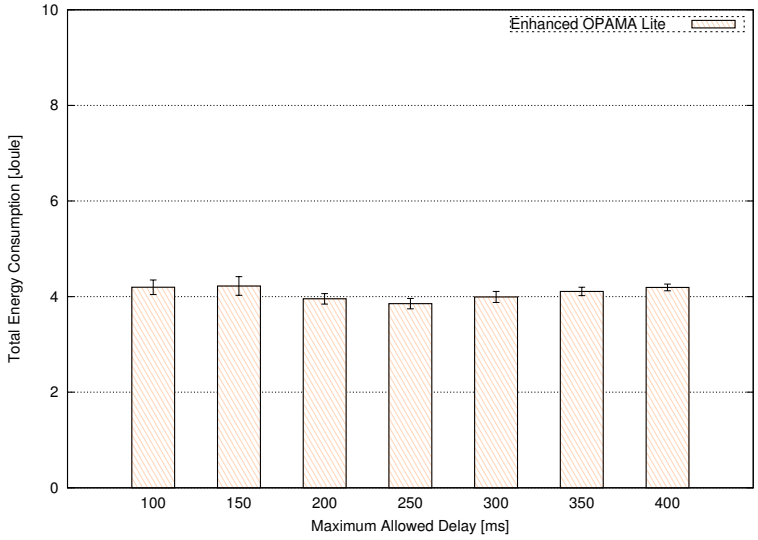


Figure 6.10: Total energy consumed by the IEEE 802.11 interface using the OPAMA lite mechanism with different maximum allowed delays.

AP queues are also increased, however, the station still needs to poll every pending packet. Therefore, the time that the station is in the sleep state does not increase substantially to reduce the energy consumption.

Figure 6.11 presents the one way delay (y-axis) when using the OPAMA lite mechanism, with different maximum allowed delays (x-axis).

The results show that the delay increases when the maximum allowed delay increases, except between the maximum allowed delays which represent a multiple of a *Beacon* interval (i.e. 100, 200, 300 and 400 ms) and the next maximum allowed delay (i.e. 150, 250 and 350 ms). This happens due to the OPAMA design. The core algorithm runs before the AP sends a *Beacon* frame, at each *Beacon* interval (100 ms), and one of the conditions to inform the station from pending data is defined by: $(timeInQueue + timeNextBeacon) > mad$. Since in the tests performed, there are always traffic and the time until the next *Beacon* is always 100 ms, a maximum allowed delay of an additional 50 ms compared with the previous maximum allowed delay, which is a multiple of a *Beacon* interval, will not make much difference. In fact, the 50 ms will only change the OPAMA lite behavior in case the packets are buffered only after the middle of the *Beacon* interval.

Furthermore, despite the fact that the OPAMA lite mechanism is able to keep the packets buffered longer, depending on the maximum allowed delay, this maximum is only granted for 75% of the packets. This occurs due to the polling phase which, as stated

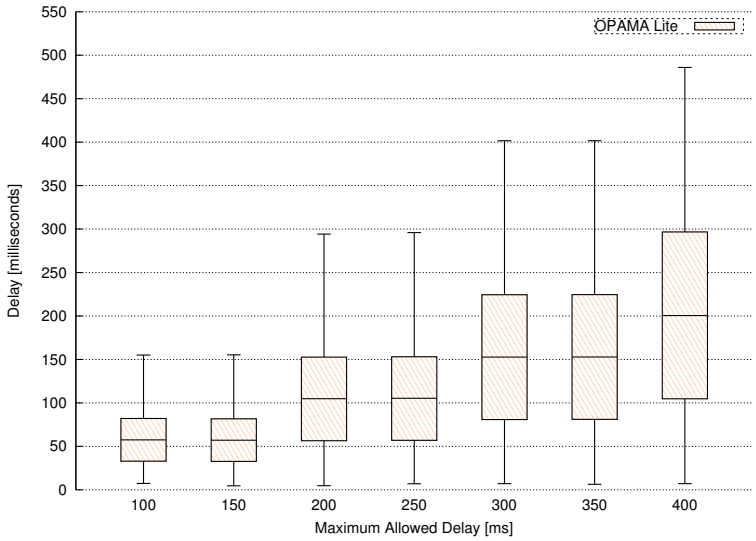


Figure 6.11: One way delay introduced by the OPAMA lite mechanism with different maximum allowed delays.

before, reduces the throughput of the transmitted packets.

The above results allowed the OPAMA lite validation regarding the implementation of the core algorithm and also the management of the buffered queues. However, the results showed that the additional delay tolerated by the stations does not always allow a reduction of the energy consumption. This occurs due to the polling phase that reduces the throughput. By adding more delay, the station is able to sleep longer, however, when the station is awake, it will need to remain in this state more time since there are more packets to poll. Additionally, the packet losses were also analyzed, however, the results showed no differences by adding more delay due to the lower transmission rate selected and, therefore, are not presented.

6.3 Enhanced OPAMA Lite

This section proposes the enhanced OPAMA lite mechanism. The enhanced OPAMA lite is build on top of the lite version of OPAMA improving it to avoid the polling phase. Therefore, the enhanced version will use the OPAMA lite core algorithm to decide if the packets must be sent. However, this mechanism requires modifications in the Android device and the design was changed to increase the OPAMA lite performance.

This section is structured as follows: Subsection 6.3.1 describes the design of the enhanced OPAMA lite, next, in Subsection 6.3.2 the implementation will be discussed and, finally, Subsection 6.3.3 presents the experimental evaluation.

6.3.1 Design

The enhanced OPAMA was designed to meet one main goal: overcome the limitations of OPAMA lite and Legacy-PSM caused by the polling phase.

Figure 6.12 shows a simplified interaction example of a station using the proposed enhanced OPAMA lite. As usually, the AP buffers packets for a station in sleep. The station wakes up to receive each *Beacon* frame which informs, in the TIM, if there is any pending data. At this point, the enhanced OPAMA lite behavior is the same of the OPAMA lite and it also uses Algorithm 2 to decide if the pending data must be sent or hidden. However, to avoid the polling phase, this mechanism uses a custom packet that informs the station if it should operate in Legacy-PSM or No-PSM mode.

Before the first *Beacon* frame, the AP verifies, based on Algorithm 2, if the queued packets must be sent. In order to avoid the polling phase, the AP adds a custom packet at the beginning of the pending frames queue. This packet will inform the station to change its mode to No-PSM. Since this packet is sent first, the station wakes up and does not need to send more *PS-Poll* frames to receive the other packets. To change the power mode, the station sends a *Null Data* frame to the AP that sends back an ACK, allowing the station to switch the current power mode.

After changing to No-PSM mode, the station is able to receive the buffered packets without polling. However, to avoid that the station remains in No-PSM after receiving the packets, the AP adds another custom packet to the pending queue. This time, the packet is added at the end of the queue to inform the station that it can go back to Legacy-PSM after receiving all the packets. Again, the station is only allowed to change its power mode after sending a *Null Data* frame and receiving an ACK.

Although this mechanism avoids the polling phase, the accomplishment of this goal is different from the original OPAMA proposed by Bernardo et al. This is done due

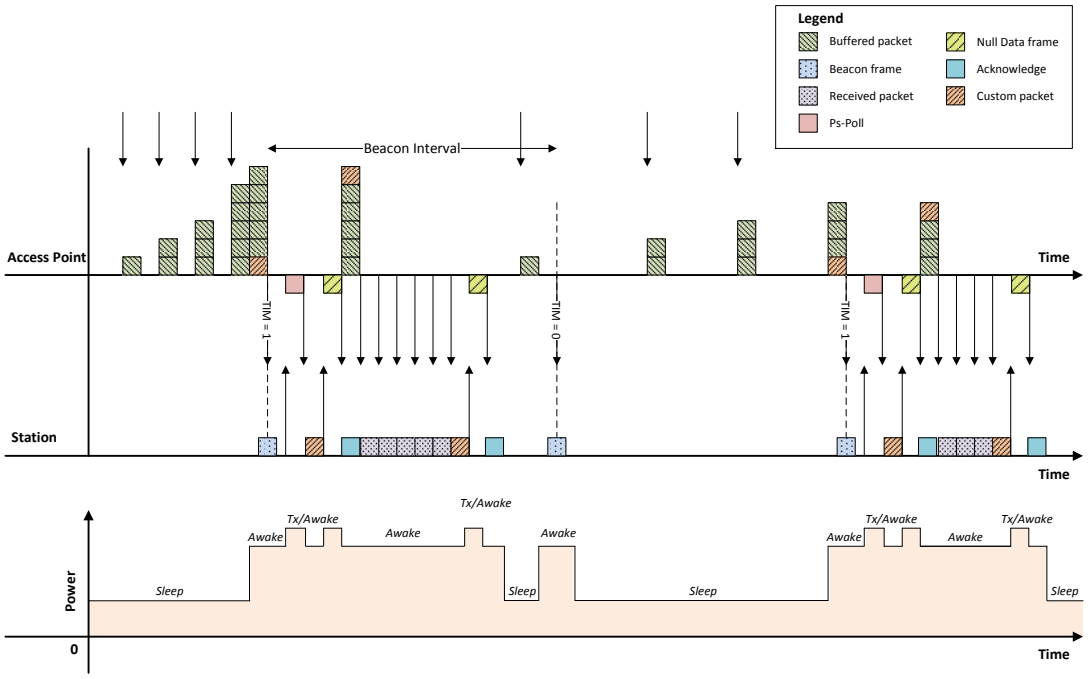


Figure 6.12: Simplified operations of the enhanced OPAMA lite and respective energy cost.

to a set of restrictions regarding the implementation of this mechanism in the Android device which will be further discussed in Subsection 6.3.2, on implementation.

When sending the second *Beacon* frame, the AP decides that the packets must not be sent. Therefore, the TIM in the *Beacon* is changed in order to hide the pending packets information. The station enters in sleep and wakes up to receive the last *Beacon* which informs that pending packets are waiting to be sent, leading to the same behavior described in the first *Beacon*.

The depicted power shows that compared with the OPAMA lite version, it is possible to reduce the energy by avoiding the polling phase. Moreover, by changing the wireless interface between the Legacy-PSM and No-PSM modes, it is possible to sleep when there are no packets to be sent and stay constantly awake when there are packets to receive, resulting in higher performances. However, when compared with the original OPAMA, there is also a higher energy consumption since there is the need to send extra packets to inform the station to change the power mode and aggregation is not performed.

6.3.2 Implementation

The implementation of the enhanced OPAMA lite was performed in the AP and also in the Android device. Concerning the implementation in the AP, it was used the same netbook with the same configurations as presented in Subsection 6.2.4. Moreover, the full implementation was conducted in the *mac80211* kernel module. The Android part was implemented as an external kernel module which communicates with the EXPoSE framework proposed in Chapter 5. The Android device used was the LG P990 already described in Chapter 4.

To implement the enhanced OPAMA mechanism in the AP, the OPAMA lite implementation was used as a base since the algorithm to decide if the packets must be sent is the same in both mechanisms.

In order to avoid the polling phase of the OPAMA lite and also of the Legacy-PSM, a mechanism was created to inform the station that it should change the power mode. As opposed to the original OPAMA, that uses the information of the TIM in the *Beacon* frame to sleep or stay awake, the enhanced OPAMA lite implementation sends an extra packet to inform the station to change the mode implicitly. This needs to be done because of strong restrictions in the drivers and the hardware of the mobile device as described next.

The wireless driver used by the LG P990 is the "*bcm4329*" [Broadcom, 2014], a common driver used in many mobile devices. However this driver belongs to the *Full-MAC* drivers family which means that the MAC sublayer Management Entity (MLME) is implemented at the hardware/firmware level. Therefore, all the control of the power management modes of the wireless could not be changed. This invalidates the possibility of changing the Legacy-PSM mode to stay awake when the TIM is 1 and sleep when it is set to 0. Moreover, the *Beacon* frames management is also performed at the hardware/firmware level. Therefore, it is not possible to parse the *Beacon* frames to get the information of the TIM and use the EXPoSE framework to change the IEEE 802.11 power mode to sleep or awake the wireless interface.

The solution found was to create a custom packet that implicitly informs the station to change the mode and to set the wireless interface into sleep or awake using the EXPoSE framework. Figure 6.13 represents the custom packet created to inform the station which one of the IEEE 802.11 power management modes must be used.

The created packet is an Ethernet packet to avoid the higher layer overheads presented, for example, in UDP and TCP. The packet is composed by five fields, namely *Destination Address*", "*Source Address*", "*Ethernet Type*", "*Sequence Number*" and

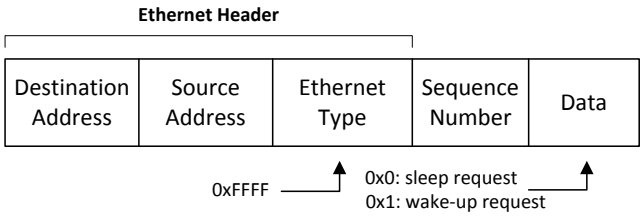


Figure 6.13: Custom Ethernet packet to control the IEEE 802.11 modes.

"*Data*" payload. The first three fields build the Ethernet header and represent the MAC address of the destination, source address and also the Ethernet type. In order to differentiate this packet from others, the Ethernet type was set with a reserved type (0xFFFF). Despite of the fact that this Ethernet type should not be used in a production stage, it is the simplest way to differentiate this packet without compromising the implementation and future evaluation of the algorithm. In fact, this technique was also used to validate other works, such as the DozyAP proposed by Han et al. [Han et al., 2012].

The "*Sequence Number*" is a number that identifies the packet and is provided by the hardware. By setting a flag in the structure that contains the information of the packet, the hardware is informed to fill the "*Sequence Number*" after sending the packet.

The "*Data*" payload contains only one bit which could be 0x0 or 0x1. The 0x0 value informs the station that it must sleep and the 0x1 value informs that the station must wake up.

In order to receive these custom packets and change the power mode of the IEEE 802.11 interface, an external kernel module was developed in the Android device. This module is loaded when the mobile device turns on and exits when the device turns off. The module acts as a hook which filters all the packets with the Ethernet type equal to 0xFFFF, before they reach the MAC stack. When a packet is received, the module parses the "*Data*" payload and verifies if it is 0x0 or 0x1. If the payload is 0x0, the module informs the EXPoSE framework to use the Legacy-PSM mode, allowing the station to stay in sleep. On the other hand, when the value is 0x1, the kernel module sends a message to the EXPoSE framework to set the wireless interface to No-PSM, allowing the station to stay awake to receive the pending data.

6.3.3 Experimental Evaluation

The enhanced OPAMA lite mechanism was evaluated in the same testbed as the OPAMA lite, using the same AP and configurations.

The performed tests aim to study the performance of the enhanced OPAMA lite mechanism, regarding its energy consumption and QoS in the presence of Continuous Media Applications. Therefore, the conducted tests were performed under the same scenario of the OPAMA lite mechanism and the obtained results will be compared with the baseline results previously analyzed in Subsection 6.2.4.

In order to understand the impact of the transmission rate in the enhanced OPAMA lite mechanism, the packet size was fixed at 1000 bytes and the transmission rate was ranged between 50 and 250 packets per second, as in the previous tests.

Figure 6.14 depicts the total energy consumption by the IEEE 802.11 interface (y-axis), while using the enhanced OPAMA lite mechanism, with a maximum allowed delay of 100 ms. The energy consumption by the wireless interface operating in the Adaptive-PSM mode is also presented in order to compare the performance of the enhanced OPAMA lite mechanism with the most used power saving mode implemented in Android devices.

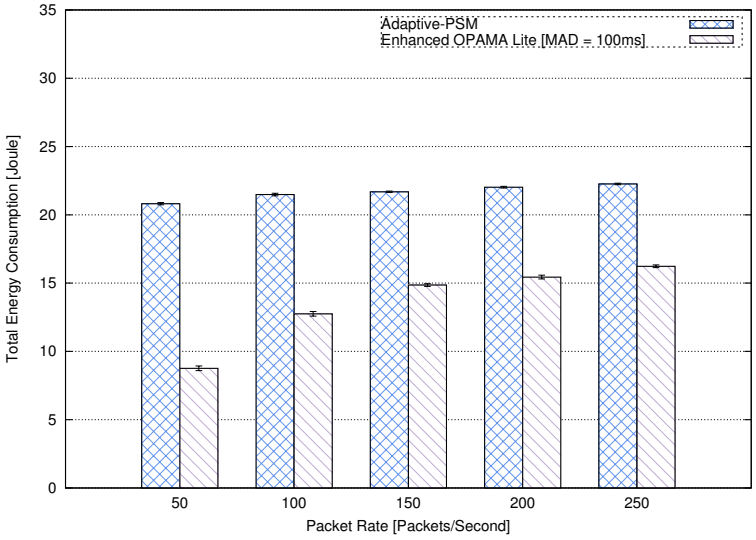


Figure 6.14: Comparison between the total energy consumed by the IEEE 802.11 interface using the enhanced OPAMA lite mechanism and the Adaptive-PSM mode with different transmission rates.

The presented results show that the enhanced OPAMA lite mechanism is clearly more energy friendly when compared with the Adaptive-PSM mode. In fact, in all the studied

rates, the enhanced OPAMA lite consumes less energy, whereas the energy consumption is between the 8.5 and 16.5 Joule while in the Adaptive-PSM mode is always higher than 20.5 Joule.

To further study the impact of the transmission rate in the enhanced OPAMA lite mechanism, the one way delay and the packet loss where also analyzed.

Figure 6.15 presents the one way delay in milliseconds (y-axis) when using the enhanced OPAMA lite mechanism and the Adaptive-PSM mode, for each studied transmission rates (x-axis).

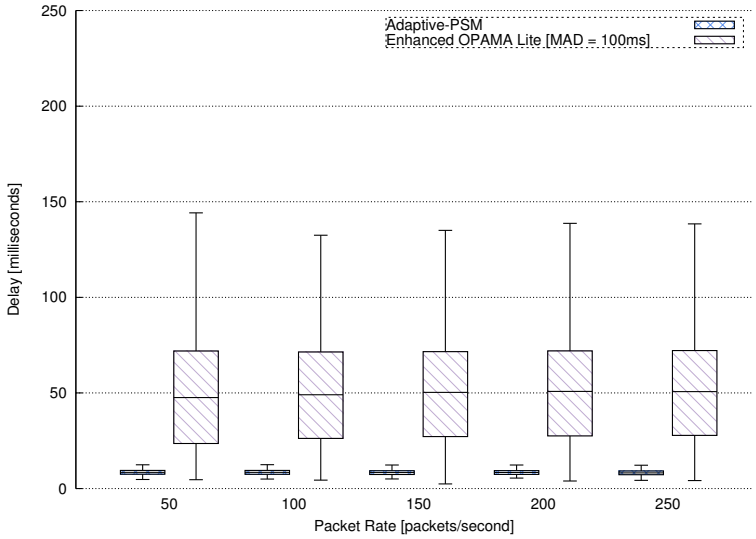


Figure 6.15: Comparison between the one way delay introduced by the enhanced OPAMA lite mechanism and the Adaptive-PSM mode with different transmission rates.

The depicted results show that the enhanced OPAMA lite has higher delays than the Adaptive-PSM mode. These results are expected since when the Adaptive-PSM is used and the traffic is continuous sent, the wireless interface is forced to remain always awake, experiencing delays always lower than 14 ms. On the other hand, the enhanced OPAMA lite is configured to allow delays with a maximum of 100 ms and, therefore, the delays introduced are higher.

Although the enhanced OPAMA mechanism is configured with a maximum allowed delay of 100 ms, the results show the maximum delay bound restricts are not guaranteed. In fact, only 75% of the packets are delivered below the maximum allowed delay. However, the maximum delays observed are always lower than 150 ms, which represents only more 50 ms than the configured maximum allowed delay by the station.

The depicted results also show that the packet rate has no impact in the delay

introduced by the enhanced OPAMA lite. These results are expected since the enhanced OPAMA lite mechanism forces the station to stay continuously awake until it has received all the buffered packets. Therefore, as stated before, when the interface is awake, the packet rate does not affect the IEEE 802.11 interface performance concerning the one way delay.

Figure 6.16 shows the percentage of the packet loss (y-axis) introduced by the enhanced OPAMA lite mechanism when receiving continuous traffic with different transmission rates (x-axis). The Adaptive-PSM mode values are not presented since, in this mode, there is no packet loss for the studied transmission rates.

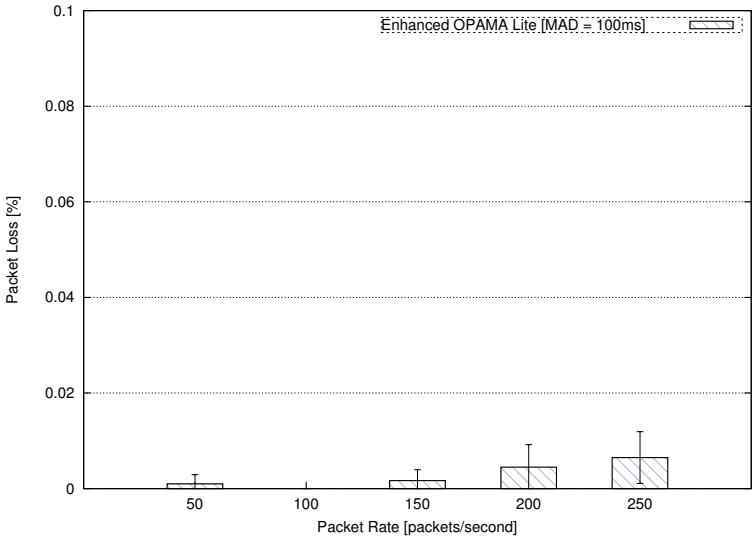


Figure 6.16: Packet loss introduced by the enhanced OPAMA lite mechanism with different transmission rates.

As expected, the enhanced OPAMA lite mechanism only introduces negligible delays, since the wireless interface is forced to remain continuously awake when the traffic must be received by the station. In fact, the percentage of packet loss is always lower than 0.02%, which represents a percentage that does not affect the QoS [Y.1541, 2011] neither the energy consumption. Additionally, it is visible that while the transmission rate increases, the packet loss tends to increase too.

In order to investigate the impact of the maximum allowed delay in the enhanced OPAMA lite behavior, the packet size was fixed at 1000 bytes and the transmission rate to 200 packets per second. Then, for each test, the maximum allowed delay was changed between 100 and 400 ms with steps of 50 ms. The configurations concerning the packet size and the transmission rate was selected since they can represent a packet generated

by a Continuous Media Application, such as the video streaming applications [Trestian et al., 2012].

Figure 6.17 depicts the percentage of the energy saved by the enhanced OPAMA lite mechanism (y-axis) when it is compared with the energy consumption of the wireless interface using the Adaptive-PSM mode.

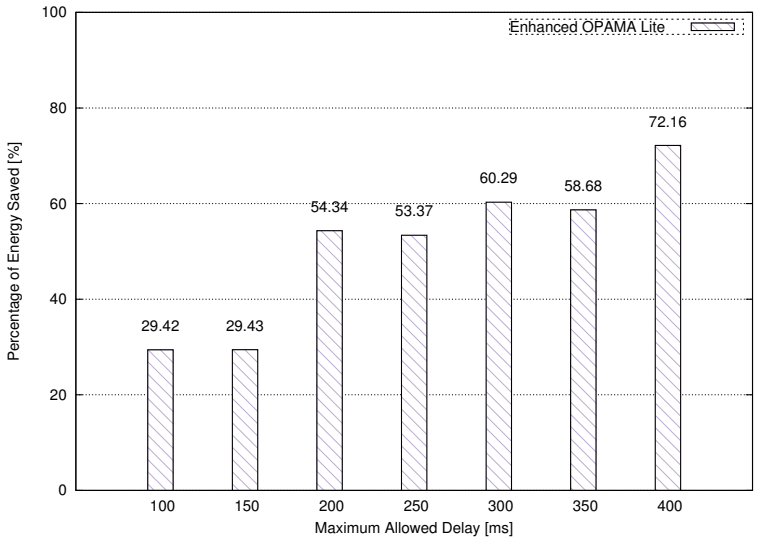


Figure 6.17: Energy savings of the enhanced OPAMA lite mechanism when compared with the Adaptive-PSM mode.

The results showed energy savings of 29.42% for the lowest configured maximum allowed delay (100 ms) and reach 72.16% when the maximum allowed delay is increased to 400 ms. Therefore, it is clear the benefit of employing the enhanced OPAMA lite mechanism in the presence of Continuous Media Applications, concerning the energy consumption.

Moreover, as already stated and explained before, it is also visible that the intermediate values for the maximum allowed delay (i.e. 150, 250 and 350 ms) do not change the performance of the enhanced OPAMA lite mechanism, although there are some energy fluctuations which do not exceed the 1.5%.

Figure 6.18 presents the delay (y-axis), introduced by the enhanced OPAMA lite mechanism, for the different configured maximum allowed delays.

By analyzing the depicted results, it is possible to conclude that the one way delay increases when the maximum allowed delay is increased. Therefore, these results show a relationship between the delay introduced and the energy consumption, i.e. the higher is the tolerated delay, the lower is the energy consumption.

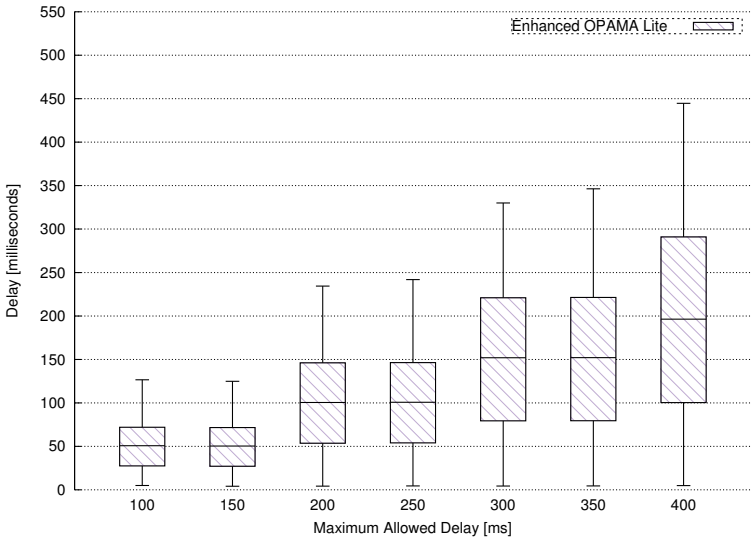


Figure 6.18: One way delay introduced by the enhanced OPAMA lite mechanism with different maximum allowed delays.

Furthermore, the obtained results show that only for the maximum allowed delays of 150, 250 and 350 ms it is possible to guarantee the delay restrictions. For the other studied maximum allowed delays, the delivery of the packets incur in extra delays always lower than 50 ms.

To complete the QoS analysis, Figure 6.19 presents the percentage of the packet loss (y-axis) for the different maximum allowed delays.

The results show that the percentage of packet loss introduced by the enhanced OPAMA lite is very low (below 0.02%) and have slight variations. Therefore, it is possible to conclude that this mechanism does not affect the QoS by introducing non negligible losses [Y.1541, 2011], as opposed to what happens in the Legacy-PSM mode.

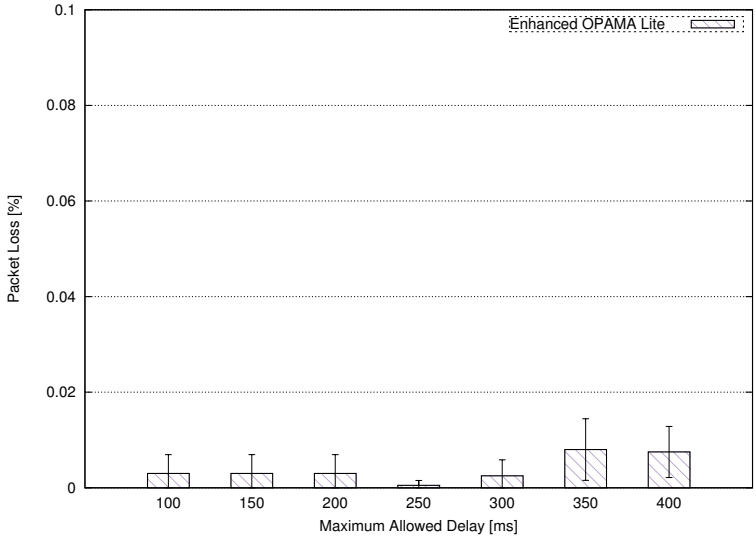


Figure 6.19: Packet loss introduced by the enhanced OPAMA lite mechanism with different maximum allowed delays.

The tests concerning the impact of the maximum allowed delay, allowed to observe a clear benefit of using the enhanced OPAMA lite approach in the presence of Continuous Media Applications. This mechanism is able to reduce significantly the energy consumption of the wireless interface, while guaranteeing the maximum delay restrictions of the station most of the times, while in others only adds more 50 ms of the expected delay by the end-users.

6.4 Summary

This chapter described the original OPAMA proposed by Bernardo et al. and presented two mechanisms, developed in this thesis for Android devices, based on the original mechanism, named by OPAMA lite and enhanced OPAMA lite. Both mechanisms presented were discussed concerning their design implementations and also the differences from the original version.

The OPAMA lite experimental results showed that this mechanism is not able to create a good trade-off between the energy consumption and the extra delay introduced. Moreover, it was also observable that the maximum allowed delay configured for the station is only granted for 75% of the packets.

Despite the fact that the OPAMA lite results do not show improvements when comparing this mechanism with the Legacy-PSM mode, the experimental evaluation allowed

the validation of the simplified OPAMA core algorithm in the AP. Therefore, it was possible to validate the implementation which extends the default Legacy-PSM behavior, by allowing the end-users to configure how long the packets remain in the AP before being transmitted to the station.

Furthermore, since the OPAMA lite represents a solution where only the AP needs to be modified, its implementation can be extended in the future, without modifying the station, to reduce energy consumption in the presence of Continuous Media Applications. For example, adding an aggregation mechanism will allow to reduce the transmission rate and consequently the polling phase responsible for the mechanism weak performance, therefore, the trade-of between the energy consumption and the QoS will be increased.

The results regarding the enhanced OPAMA lite showed a clear benefit of using this mechanism to reduced the energy consumption in the presence of Continuous Media Applications. In fact, as opposed to the current power modes implemented in Android devices, allows to achieve a good trade-off between the energy consumption and the end-user expectations.

The analyzed results showed energy reductions between 29.42% and 72% when compared with the Adaptive-PSM mode. Moreover, the enhanced OPAMA lite performance showed that it is possible to ensure the delay bound restrictions when the maximum allowed delay is set to 150, 250 and 350 ms. For the other maximum delays studied, the mechanism only adds extra delays always lower than 50 ms, while ensuring that 75% of the packets are delivered below the maximum delay configured.

When comparing the enhanced OPAMA lite with the Legacy-PSM mode or the OPAMA lite mechanism, it is noticeable that avoiding the polling phase has a strong impact in the wireless interface performance. In fact, by reducing the polling phase, as opposed to what happens with the Legacy-PSM mode, it is possible to reduce significantly the energy consumption while ensuring the end-user expectations, namely the maximum delay tolerated, without introducing packet losses.

Chapter 7

Project Management

This chapter describes the first and second semester activities, regarding the work plan of the Master Thesis presented in this document.

The first section presents the work plan for the first semester and the real work conducted. The second section describes the proposed work plan concerning the second semester and the effective work plan performed.

7.1 First Semester

Figure 7.1, illustrates the initial proposed plan for the first semester. Since the predicted tasks experienced delays and changes, the progress of the project took a different direction from the initial plan. Therefore, the effective schedule of the work plan is presented in Figure 7.2.

ID	Task Name	Start	Finish	Duration
1	**Study of state-of-the-art**	**9/16/2013**	**11/1/2013**	**35d**
2	Greening Wireless Communications	9/16/2013	9/20/2013	5d
3	IEEE 802.11	9/23/2013	10/11/2013	15d
4	Video Streaming in Android Devices	10/14/2013	11/1/2013	15d
5	**Android preliminary setup**	10/1/2013	10/2/2013	2d
6	**Evaluate IEEE 802.11 power saving mechanisms in Android devices**	10/1/2013	10/25/2013	19d
7	**Android IEEE 802.11 power control mechanism**	10/21/2013	11/29/2013	30d
8	**Tests and Analysis**	12/2/2013	12/30/2013	21d
9	**Writing Report**	10/21/2013	1/28/2014	72d

Figure 7.1: Project plan first semester - Initial proposed schedule.

ID	Task Name	Start	Finish	Duration
1	**Study of state-of-the-art**	**9/16/2013**	**11/1/2013**	**35d**
2	Greening Wireless Communications	9/16/2013	9/20/2013	5d
3	IEEE 802.11 MAC layer – Power save modes	9/23/2013	10/4/2013	10d
4	IEEE 802.11 energy efficiency mechanisms	10/7/2013	10/25/2013	15d
5	IEEE 802.11 energy assessment	10/28/2013	11/1/2013	5d
6	**Android preliminary setup**	11/4/2013	11/4/2013	1d
7	**Android study**	**11/5/2013**	**11/21/2013**	**13d**
8	Android architecture	11/5/2013	11/6/2013	2d
9	IEEE 802.11 implementation in Android	11/8/2013	11/21/2013	10d
10	**Android IEEE 802.11 power control mechanism**	**11/11/2013**	**12/26/2013**	**34d**
11	Architecture specification	11/11/2013	11/12/2013	2d
12	"wpa_cli" cross-compilation	11/12/2013	12/2/2013	15d
13	New architecture specification	12/4/2013	12/5/2013	2d
14	Prototype Implementation	12/6/2013	12/9/2013	2d
15	Kowalski Kernel compilation	12/6/2013	12/26/2013	15d
16	**Tests and analysis**	**12/24/2013**	**1/21/2014**	**21d**
17	Testbed configuration	12/24/2013	12/30/2013	5d
18	Specification	1/2/2014	1/6/2014	3d
19	Execution	1/7/2014	1/16/2014	8d
20	Results and analysis	1/10/2014	1/21/2014	8d
21	**Writing report**	11/4/2013	1/28/2014	62d

Figure 7.2: Project plan first semester - Effective schedule.

The first performed task was to study the problematic of the green wireless communications, which was followed by the study of the IEEE 802.11 technology, with a focus in the power saving techniques concerning the MAC layer. After understanding the basic concepts, the focus has become the study of the existent mechanisms in the literature, regarding the energy efficiency in IEEE 802.11. Moreover, the study of methodologies to assess the energy consumption in IEEE 802.11 was also conducted.

The study of the state-of-the-art was followed by the study of the used platform, the Android OS. This study was divided in two parts, the general architecture of Android and the architecture of the IEEE 802.11 implementation in Android.

After the knowledge about the Android OS and the implementation of the IEEE 802.11 technology in this platform has been acquired, a task concerning the development of a mechanism to control the IEEE 802.11 sleep periods was started. This task was done in two loops and remained unfinished. Therefore, the conclusion of this task was moved to the second semester work plan. The two loops of the task represent the first approach in the specification of the mechanism, that was discarded, since various problems

were identified which made the development of the mechanism unfeasible. Therefore, a second specification of the mechanism was created and the initial steps regarding its implementation were conducted. In both loops, the compilation of the "*wpa_cli*" and the Android kernel have led to an additional effort in the planned work. These unexpected problems represent the highest delays occurred during the first semester.

A preliminary set of tests and its analysis was also conducted. This task was divided by the testbed configuration, the test specifications, the test execution and the analysis of the results.

Finally, the task regarding the writing of the report was developed along the semester, with the major effort at the end.

7.2 Second Semester

The proposed plan for the second semester is presented in Figure 7.3. As in the first semester, the tasks proposed experienced schedule adjustments due to the introduction of new tasks and the modification of others. Moreover, some tasks took longer than expected, while others were performed in less time. Therefore, Figure 7.4 depicts a Gantt chart which represents the effective schedule of the second semester.

ID	Task Name	Start	Finish	Duration
1	**IEEE 802.11 control sleep Android framework**	**1/29/2014**	**3/4/2014**	**25d**
2	Kowalski Kernel modifications	1/29/2014	2/18/2014	15d
3	Framework implementation	2/19/2014	3/4/2014	10d
4	**OPAMA lite**	**3/5/2014**	**5/13/2014**	**50d**
5	Implementation	3/5/2014	3/18/2014	10d
6	Test specification and execution	3/19/2014	4/29/2014	30d
7	Analysis	4/23/2014	5/13/2014	15d
8	**Enhanced OPAMA lite**	**5/14/2014**	**7/2/2014**	**36d**
9	Specification	5/14/2014	5/20/2014	5d
10	Implementation	5/21/2014	6/3/2014	10d
11	Test specification and execution	6/4/2014	6/24/2014	15d
12	Analysis and validation	6/23/2014	7/2/2014	8d
13	**Writing report**	4/23/2014	7/7/2014	54d

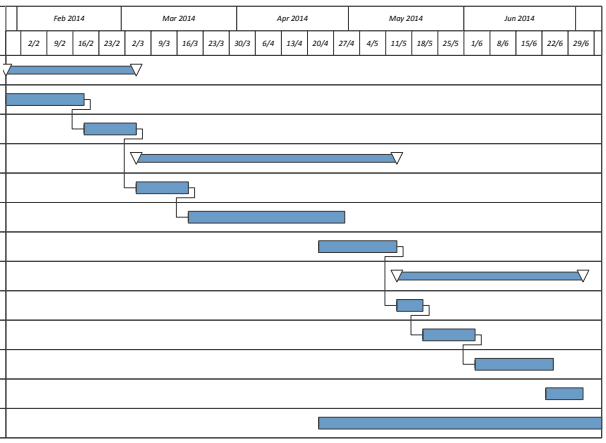
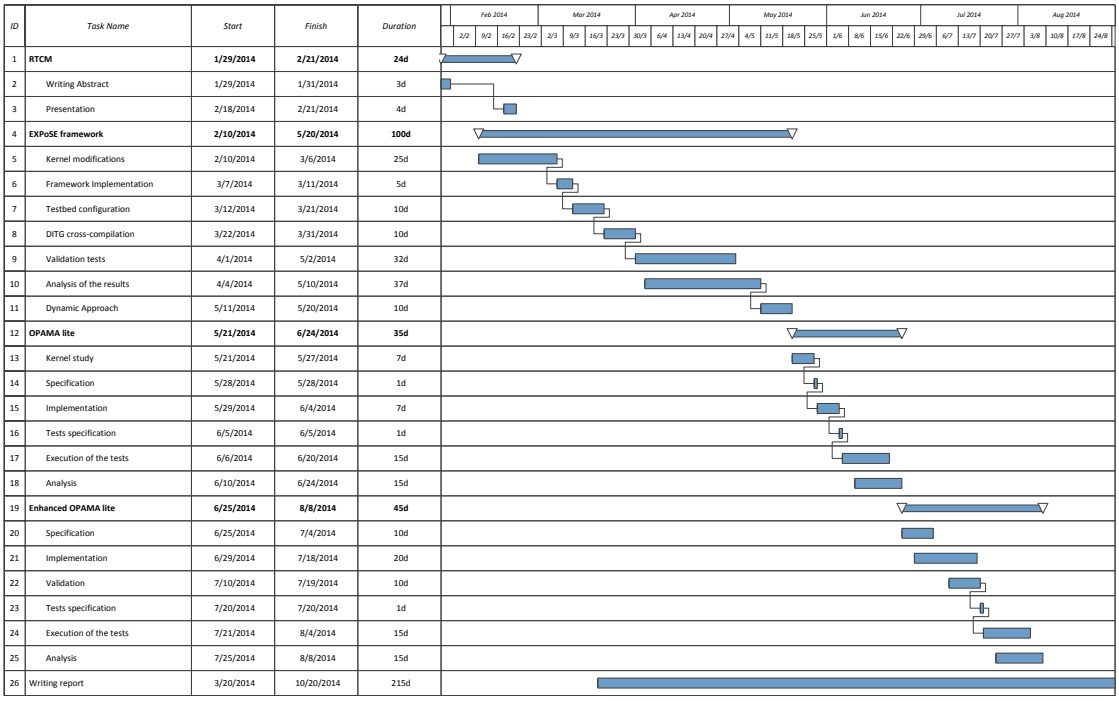


Figure 7.3: Project plan second semester - Initial proposed schedule.



ID	Task Name	Start	Finish	Duration
1	**RTCM**	**1/29/2014**	**2/21/2014**	**24d**
2	Writing Abstract	1/29/2014	1/31/2014	3d
3	Presentation	2/18/2014	2/21/2014	4d
4	**EXPoSE framework**	**2/10/2014**	**5/20/2014**	**100d**
5	Kernel modifications	2/10/2014	3/6/2014	25d
6	Framework Implementation	3/7/2014	3/11/2014	5d
7	Testbed configuration	3/12/2014	3/21/2014	10d
8	DITG cross-compilation	3/22/2014	3/31/2014	10d
9	Validation tests	4/1/2014	5/2/2014	32d
10	Analysis of the results	4/4/2014	5/10/2014	37d
11	Dynamic Approach	5/11/2014	5/20/2014	10d
12	**OPAMA lite**	**5/21/2014**	**6/24/2014**	**35d**
13	Kernel study	5/21/2014	5/27/2014	7d
14	Specification	5/28/2014	5/28/2014	1d
15	Implementation	5/29/2014	6/4/2014	7d
16	Tests specification	6/5/2014	6/5/2014	1d
17	Execution of the tests	6/6/2014	6/20/2014	15d
18	Analysis	6/10/2014	6/24/2014	15d
19	**Enhanced OPAMA lite**	**6/25/2014**	**8/8/2014**	**45d**
20	Specification	6/25/2014	7/4/2014	10d
21	Implementation	6/29/2014	7/18/2014	20d
22	Validation	7/10/2014	7/19/2014	10d
23	Tests specification	7/20/2014	7/20/2014	1d
24	Execution of the tests	7/21/2014	8/4/2014	15d
25	Analysis	7/25/2014	8/8/2014	15d
26	Writing report	3/20/2014	10/20/2014	215d

Figure 7.4: Project plan second semester - Effective schedule.

The main tasks performed during the second semester, present in Figure 7.4, are described next:

- **RTCM:** This was the first task performed in the second semester where an abstract was submitted to the "*Rede Temática de Comunicações Móveis*" (RTCM) [RTCM, 2014]. The abstract gathered the work done during the first semester and can be found in Appendix A. This task ended with the presentation of the work performed in the first semester at the *18º Seminário da RTCM*.

- **EXPoSE framework:** This task was started in the first semester, where the study of the Android kernel was conducted and a specification of the framework was proposed. Therefore, in the second semester, the EXPoSE framework development started with the modifications of the kernel, followed by the implementation of the framework which includes the implementation of the EXPoSE service and the integration between the framework components. After developing the EXPoSE service, the testbed was configured. This sub-task took longer than expected since it was necessary to find a solution to sync the testbed machines without compromising the energy consumption by the Android device. Then, the EXPoSE fra-

mework was validated in the testbed and, at the same time, the obtained results were analyzed. This task ended with the implementation of a dynamic mechanism to reduce the energy consumption in the presence of Continuous Media Applications which remained unfinished due to the time constraints, concerning the work proposed for this thesis. As shown by the differences between the Gantt chart depicted in Figure 7.3 and the one in Figure 7.4 this was the task that suffered more modifications. This happened due to two reasons: first, the implementation of the kernel modifications were delayed due to the RTCM submission, and, as opposed to the first plan, the EXPoSE framework also included functionalities to reduce energy consumption in the presence of Continuous Media Applications, taking into account the end-user expectations. The validation of these functionalities resulted in the publication of a paper which can be found in Appendix A.

- **OPAMA lite:** This task was also modified concerning its initial goals, since it was decided to conduct the OPAMA implementation in two steps: a first solution that only needed modifications in the AP, the OPAMA lite, and an enhancement of this mechanism which represents a solution where the AP and the Android device needed modifications, the enhanced OPAMA lite. Therefore, the OPAMA lite mechanism main goal was to allow the implementation and validation of a simplified version of the OPAMA mechanism in the AP.

- **Enhanced OPAMA lite:** After the implementation and validation of the OPAMA lite mechanism, the enhanced OPAMA lite was created as an optimization of the OPAMA lite performance. However, due to the hardware and driver restrictions, this version could not be implemented as the original OPAMA. Therefore, it was specified and implemented a mechanism which overcomes the hardware and driver restrictions and enhances the OPAMA lite mechanism to avoid the polling phase, as done in the original OPAMA. Due to the complexity of this task, the estimated time for its conclusion was not enough. Moreover, it was also necessary to include a new sub-task to validate the mechanism, since, its implementation is different from the original.

- **Writing of the report:** This task represents the writing of the final report. All the work performed, the results and the final conclusions are compiled in this document. This task was performed in this thesis during the first and second semesters.

As can be observed in the Gantt diagrams, the estimated project plan for this Mas-

ter thesis had to be rescheduled along time. These modifications were made due to many reasons, whereas the most relevant was the complexity of the proposed work. In fact, the highest delays presented can be explained due to the complexity of the kernel implementations and the complexity of porting tools to the Android device. However, despite of the complexity, the adjustments in the work plan allowed to achieve the main objectives of this thesis.

Chapter 8

Final Considerations

The IEEE 802.11 technology is one of the most used wireless interfaces to give Internet access in many devices, however, it is also an energy consuming component. Therefore, increasing the lifetime of battery dependent devices, while using the IEEE 802.11 technology, must be a concern.

In this context, the work performed in this thesis addressed the IEEE 802.11 technology in Android devices, regarding its energy consumption and network performance in the presence of Continuous Media Applications.

The study of the IEEE 802.11 technology was the first task performed in this thesis and included the analysis of the IEEE 802.11 standard, the study of the Medium Access Control (MAC) layer and also the power management modes developed for this technology.

Concerning the state of the art for this work, various mechanisms presented in the literature, which aim to reduce the energy consumption of the IEEE 802.11 interface were analyzed. Through this analysis, it was possible to understand that most of the proposals achieve energy savings by keeping the wireless interface in sleep for more time. This is done by holding the packets in queues after sending the packets to the destination address or by aggregating the packets to be sent together. In both approaches, the reduction of the energy consumption has a cost of adding delay. In fact, adding extra delays is usually necessary to make it possible to reduce the wireless interface energy consumption. Therefore, the challenge becomes the balance between the energy consumption by the IEEE 802.11 interface and its network performance.

Despite of the already existing mechanisms in the literature, the energy efficiency focused on Continuous Media Application has not been much explored. Furthermore, the characterization of the IEEE 802.11 interface, performed in this thesis, showed that the current power saving modes developed for Android devices do not bring any improvement

in the presence of continuous traffic. In fact, the Legacy-PSM mode defined in the IEEE 802.11 standard introduces a high percentage of packet losses in the presence of Continuous Media Applications and can add excessive delays. On the other hand, the Adaptive-PSM mode, widely used in Android devices, does not introduce extra delays or losses when compared with the No-PSM mode. However, it is not able to reduce the energy consumption of the wireless interface since, due to the nature of continuous traffic, this mode sets the wireless interface to be continuously awake as in the No-PSM mode.

Furthermore, the existent power saving modes in Android devices and also most of the mechanisms presented in the literature, do not take into account the end-user requirements. To overcome this limitations, the work performed in this thesis allowed the proposal of the EXPoSE framework. This framework extends the current IEEE 802.11 technology implementation in Android devices, in order to give full control of the wireless interface to the end-users. Moreover, the EXPoSE framework provides two approaches to reduce energy in the presence of Continuous Media Applications, the pattern-based approach and the maximum allowed delay approach. The performed results to validate the EXPoSE framework showed energy savings up to 44%, comparing with the Adaptive-PSM mode, with the cost of maximum delays below the 240 ms when the pattern-based approach is used. The results concerning the maximum allowed delay approach showed energy savings of 23.53% when comparing with the Adaptive-PSM mode, while keeping the end-user expectations in the studied scenarios. In both approaches packet losses that affect the QoS were not noticeable.

Besides the development of the EXPoSE framework, this thesis also presented the OPAMA lite and the enhanced OPAMA lite mechanisms which were based on the OPAMA solution, proposed by Bernardo et al. [Bernardo et al., 2013]. The OPAMA lite allowed to extend the default Legacy-PSM mode, by changing the AP in order to enable the end-users to configure a maximum tolerated delay. Therefore, the AP uses this value to keep the packets buffered for longer, allowing the stations to sleep even if the AP has buffered packets to it. Despite that the results showed no significantly energy reductions when the OPAMA lite is applied, the experimental evaluation allowed to validate the AP implementation which can be used in the future to enhance this mechanism by only changing the AP behavior.

The enhanced OPAMA lite was developed to overcome the OPAMA lite problems, namely the polling phase. The results showed a clear benefit of using this mechanism since it is possible to reduce the IEEE 802.11 interface energy consumption between 29.42% and 72.16% while adding delays between 150 and 450 ms without incurring

in significant losses. Therefore, the analysis of the enhanced OPAMA lite performed showed that, as opposed to the current power saving mechanisms in Android devices, it is possible to achieve a good trade-off between the energy consumption and the end-user expectations, in the presence of Continuous Media Applications.

Despite of the fact that the enhanced OPAMA lite allows significant energy savings in the presence of continuous traffic, the experimental evaluation showed that the maximum allowed delay configured is not always achieved. Therefore, the future work should be focused on increasing this mechanism performance so that the maximum allowed delay is guaranteed. The implementation of an aggregation mechanism should also be considered, since by aggregating the packets the transmission rate is reduced leading the station to stay in the sleep state for long time periods, reducing the energy consumption. Furthermore, the evaluation of this type of mechanisms with real traffic, such as VoIP or video, becomes important as well as the QoE assessment which will allow to understand the real impact of these mechanisms in the end-users perception.

The work conducted in this thesis showed that it is possible to overcome the existent limitations of the Android devices concerning the Continuous Media Applications, by proposing two mechanisms to extend the IEEE 802.11 implementation in order to achieve a good trade-off between the energy consumption and the end-user expectations.

References

[Amuhong, 2014] Amuhong. IEEE 802.11 architecture in Android., 2014. URL `http://blog.chinaunix.net/uid-22415790-id-3651042.html`. [Online; accessed 23-January-2014]. xi, xiii, 47, 131

[Android, 2014a] Android. Android developers, about web page., 2014a. URL `http://developer.android.com/about/index.html`. [Online; accessed 23-January-2014]. 1

[Android, 2014b] Android. Android low level system architecture., 2014b. URL `http://source.android.com/devices/`. [Online; accessed 23-January-2014]. xiii, 129

[Android, 2014c] Android. Android software stack., 2014c. URL `http://source.android.com/devices/tech/security/index.html`. [Online; accessed 23-January-2014]. xiii, 129

[Bernardo and Curado, 2012] V. Bernardo and M. Curado. A methodology for assessing video transmission energy consumption and quality. In *2012 IEEE International Conference on Communications (ICC)*, pages 6308–6313, 2012. 20, 24

[Bernardo et al., 2011] V. Bernardo, M. Curado, T. Staub, and T. Braun. Towards energy consumption measurement in a cloud computing wireless testbed. In *2011 First International Symposium on Network Cloud Computing and Applications (NCCA)*, pages 91–98, 2011. xi, 3, 19, 20, 24, 28, 29, 36

[Bernardo et al., 2013] Vitor Bernardo, Marilia Curado, and Torsten Braun. Enhancing IEEE 802.11 energy efficiency for continuous media applications. In Jean-Marc Pierson, Georges Da Costa, and Lars Dittmann, editors, *Energy Efficiency in Large Scale Distributed Systems*, Lecture Notes in Computer Science, pages 203–217. Springer Berlin Heidelberg, January 2013. ISBN 978-3-642-40516-7, 978-3-642-40517-4. URL

http://link.springer.com/chapter/10.1007/978-3-642-40517-4_17. 1, 3, 16, 22, 65, 69, 102

[Bernardo et al., 2014] Vitor Bernardo, Bruno Correia, Marilia Curado, and Torsten Ingo Braun. Towards end-user driven power saving control in android devices. In Sergey Balandin, Sergey Andreev, and Yevgeni Koucheryavy, editors, *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*, number 8638 in Lecture Notes in Computer Science, pages 231–244. Springer International Publishing, January 2014. ISBN 978-3-319-10352-5, 978-3-319-10353-2. URL http://link.springer.com/chapter/10.1007/978-3-319-10353-2_20. 3, 62

[Botta et al., 2012] Alessio Botta, Alberto Dainotti, and Antonio Pescapè. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531–3547, 2012. 31

[Broadcom, 2014] Broadcom. Broadcom bcm4329, 2014. URL http://www.broadcom.com/products/Bluetooth/Bluetooth-RF-Silicon-and-Software-Solutions/BCM4329. [Online; accessed 25-August-2014]. 85

[Cisco, 2013] Cisco. Cisco visual networking index: Global mobile data traffic forecast update, 20122017, 2013. URL http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html. 2

[Csernai and Gulyas, 2011] M'rton Csernai and A. Gulyas. Wireless adapter sleep scheduling based on video QoE: How to improve battery life when watching streaming video? In *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–6, 2011. 18, 22

[Ding et al., 2012] Ning Ding, A. Pathak, D. Koutsonikolas, C. Shepard, Y.C. Hu, and Lin Zhong. Realizing the full potential of PSM using proxying. In *2012 Proceedings IEEE INFOCOM*, pages 2821–2825, 2012. 13, 17, 22

[G.114, 2003] ITU-T G.114. ITU-T recommendation G.114. Technical report, International Telecommunication Union, 2003. 59

[Han et al., 2012] Hao Han, Yunxin Liu, Guobin Shen, Yongguang Zhang, and Qun Li. DozyAP: Power-efficient wi-fi tethering. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, page 421434, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1301-8. URL http://doi.acm.org/10.1145/2307636.2307675. 18, 22, 86

[Hostapd, 2014] Hostapd. Hostapd webpage., 2014. URL `http://hostap.epitest.fi/hostapd/`. [Online; accessed 22-January-2014]. 73

[IEEE, 1997] IEEE. IEEE standard for information technology- telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements-part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Std 802.11-1997*, pages i–445, 1997. 5

[IEEE, 1999a] IEEE. Supplement to 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: High-speed physical layer in the 5 GHz band. *IEEE Std 802.11a-1999*, pages i–, 1999a. 5

[IEEE, 1999b] IEEE. Supplement to 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: Higher-speed physical layer extension in the 2.4 GHz band. *IEEE Std 802.11b-1999*, pages i–90, 1999b. 5

[IEEE, 2003] IEEE. IEEE standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements part ii: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Std 802.11g-2003 (Amendment to IEEE Std 802.11, 1999 Edn. (Reaff 2003) as amended by IEEE Stds 802.11a-1999, 802.11b-1999, 802.11b-1999/Cor 1-2001, and 802.11d-2001)*, pages i–67, 2003. 5

[IEEE, 2005] IEEE. IEEE standard for information technologyLocal and metropolitan area networksSpecific requirementsPart 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications - amendment 8: Medium access control (MAC) quality of service enhancements. *IEEE Std 802.11e-2005 (Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003)*, pages 1–212, 2005. 5, 9, 12

[IEEE, 2007] IEEE. IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pages 1–1076, 2007. 5

[IEEE, 2009] IEEE. IEEE standard for information technology local and metropolitan area networks specific requirements part 11: Wireless LAN medium access control (MAC)and physical layer (PHY) specifications amendment 5: Enhancements for higher throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pages 1–565, 2009. 6, 12

[IEEE, 2012] IEEE. IEEE standard for information technologyTelecommunications and information exchange between systems local and metropolitan area networksSpecific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, 2012. xi, 6, 7, 8, 10

[Iperf, 2014] Iperf. Iperf tool webpage., 2014. URL `http://iperf.fr`. [Online; accessed 22-January-2014]. 31

[Kennedy et al., 2011] M. Kennedy, H. Venkataraman, and G. Muntean. Dynamic stream control for energy efficient video streaming. In *2011 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–6, 2011. 34, 35

[MAC80211, 2014] MAC80211. Linux wireless, mac80211, webpage., 2014. URL `http://wireless.kernel.org/en/developers/Documentation/mac80211`. [Online; accessed 19-August-2014]. 72

[Pyles et al., 2011] Andrew J. Pyles, Zhen Ren, Gang Zhou, and Xue Liu. SiFi: exploiting VoIP silence for WiFi energy savings insmart phones. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, page 325334, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0630-0. URL `http://doi.acm.org/10.1145/2030112.2030157`. 17, 22, 34, 35, 49

[Pyles et al., 2012] Andrew J. Pyles, Xin Qi, Gang Zhou, Matthew Keally, and Xue Liu. SAPSM: smart adaptive 802.11 PSM for smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, page 1120, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1224-0. URL `http://doi.acm.org/10.1145/2370216.2370219`. 13, 16, 22

[Rice and Hay, 2010] Andrew Rice and Simon Hay. Measuring mobile phone energy consumption for 802.11 wireless networking. *Pervasive Mob. Comput.*, 6(6):593606, December 2010. ISSN 1574-1192. URL `http://dx.doi.org/10.1016/j.pmcj.2010.07.005`. 3, 20, 24, 29

[RTCM, 2014] RTCM. "*Rede Temática de Comunicações Móveis*"., 2014. URL http://rtcm.inescn.pt/index.php?id=424. [Online; accessed 27-January-2014]. 3, 98

[Trestian et al., 2012] R. Trestian, A.-N. Moldovan, O. Ormond, and G. Muntean. Energy consumption analysis of video streaming to android mobile devices. In *2012 IEEE Network Operations and Management Symposium (NOMS)*, pages 444–452, 2012. 36, 38, 90

[Tsao and Huang, 2011] Shiao-Li Tsao and Chung-Huei Huang. Review: A survey of energy efficient MAC protocols for IEEE 802.11 WLAN. *Comput. Commun.*, 34(1):5467, January 2011. ISSN 0140-3664. URL http://dx.doi.org/10.1016/j.comcom.2010.09.008. xi, 9, 11

[Vergara and Nadjm-Tehrani, 2013a] E.J. Vergara and S. Nadjm-Tehrani. Watts2Share: energy-aware traffic consolidation. In *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, pages 14–22, 2013a. 15, 22

[Vergara and Nadjm-Tehrani, 2013b] Ekhiotz Jon Vergara and Simin Nadjm-Tehrani. EnergyBox: a trace-driven tool for data transmission energy consumption studies. In Jean-Marc Pierson, Georges Da Costa, and Lars Dittmann, editors, *Energy Efficiency in Large Scale Distributed Systems*, Lecture Notes in Computer Science, pages 19–34. Springer Berlin Heidelberg, January 2013b. ISBN 978-3-642-40516-7, 978-3-642-40517-4. URL http://link.springer.com/chapter/10.1007/978-3-642-40517-4_2. 15, 19, 24

[Vergara et al., 2013] Ekhiotz Jon Vergara, Joseba Sanjuan, and Simin Nadjm-Tehrani. Kernel level energy-efficient 3G background traffic shaper for android smartphones. pages 443–449. IEEE, July 2013. ISBN 978-1-4673-2480-9, 978-1-4673-2479-3, 978-1-4673-2478-6. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6583599. 15, 22

[Y.1541, 2011] ITU-T Y.1541. Y.1541: Network Performance Objectives for IP-Based Services. Technical report, International Telecommunication Union, 2011. URL http://www.itu.int/itudoc/itu-t/aap/sg13aap/history/y1541/y1541.html. 59, 62, 89, 91

[Yaghmour, 2013] Karim J. Yaghmour. *Embedded Android*. March 2013. URL http://shop.oreilly.com/product/0636920021094.do. 127, 128

Appendix A

This appendix presents all the accepted publications during the development of this thesis.

Towards Energy Efficient Multimedia Streaming in Mobile Devices

Bruno Correia, Vítor Bernardo, Marília Curado
bcorreia@student.dei.uc.pt, {vmbern, marilia}@dei.uc.pt

January 31, 2014

Abstract

The progress of technology allowed the Internet access wherever we are and wherever we go. Despite the emerging of new technologies that enable access to the Internet anywhere, such as the cellular networks, the IEEE 802.11 technology is still one of the most used, being present in almost every electronic device.

The mobile devices are now embedded in the daily user routines and the smarphones have an important role in this connection between people and technology. In fact, the mobile devices have been steadily growing in recent years with the Android devices reaching the milestone of 1 million devices activated every day. Since these devices are constantly used and, most of the time, connected to the Internet, its battery lifetime is becoming a concern.

The IEEE 802.11 interface is one of the components in Android devices responsible for a large part of the energy consumption. Moreover, the most implemented IEEE 802.11 power save modes in the Android devices do not bring any energy reduction when the users are watching a video transmitted over streaming. This is a critical problem, since it is expected that, in 2017, the video traffic will represent 66.5% of all mobile traffic.

The facts presented showed the importance of proposing new mechanisms to optimize the energy consumption in Android devices, in order to achieve a tradeoff between the performance and the energy consumed by the device. Therefore, this work presents the IEEE 802.11 technology, with a focus on the Media Access Control (MAC) layer and the power save mechanisms defined in the standard. Some techniques presented in the literature, which aim to optimize the energy consumption in IEEE 802.11, are discussed and the methodologies to assess the IEEE 802.11 energy consumption are presented. A measurement testbed setup is described and allows the assessment of the energy consumption in Android devices. This work ends with an analysis of a set of tests regarding the energy consumed by the IEEE 802.11 interface and the correspondent power save mode mechanisms implemented in the Android devices.

Keywords: IEEE 802.11, Energy, Video streaming, Android Operating System.

Towards End-User Driven Power Saving Control in Android devices

Vitor Bernardo[1], Bruno Correia[1], Marilia Curado[1], and Torsten Braun[2]

[1] Center for Informatics and Systems, University of Coimbra, Coimbra, Portugal
{vmbern,bcorreia,marilia}@dei.uc.pt

[2] Institute for Computer Science and Applied Mathematics,
University of Bern, Bern, Switzerland
braun@iam.unibe.ch

Abstract. During the last decade mobile communications increasingly became part of people's daily routine. Such usage raises new challenges regarding devices' battery lifetime management when using most popular wireless access technologies, such as IEEE 802.11. This paper investigates the energy/delay trade-off of using an end-user driven power saving approach, when compared with the standard IEEE 802.11 power saving algorithms. The assessment was conducted in a real testbed using an Android mobile phone and high-precision energy measurement hardware. The results show clear energy benefits of employing user-driven power saving techniques, when compared with other standard approaches.

Keywords: Energy Efficiency, Power Saving, IEEE 802.11, Android, Testbed

1 Introduction

Nowadays, wirelessly connected mobile devices are present almost everywhere at any time. Apart from other available wireless technologies, IEEE 802.11 [1] seems to be the *de-facto* standard for wireless communications, being supported in millions of devices. Although several mobile devices have Internet access through mobile operator networks, performance limitations on the support of highly demanding multimedia applications enabled novel and hybrid communication paradigms (e.g. offloading) where IEEE 802.11 plays an important role [2].

In this context, energy consumption issues of battery-supported devices need to be addressed. In particular, since the Android [3] platform is responsible for a large part of the mobile device market growth, IEEE 802.11 energy management mechanisms in this platform should be carefully investigated.

This work studies and compares, using a real testbed, the most popular IEEE 802.11 power saving techniques implemented on the Android platform. Additionally, an Android framework for Extending Power Saving control to End-users (EXPoSE) is proposed, aiming at improving the devices' energy efficiency by considering end-users demands.

The rest of this paper is structured as follows: Section 2 introduces the technology background and discusses the most significant related work. An overview of the Android platform architecture, followed by the presentation of the EXPoSE framework is given in Section 3. Section 4 describes the evaluation testbed and discusses the obtained results. Finally, Section 5 presents the conclusions.

2 Related Work

This section presents the background concerning the standard power saving mechanisms of the IEEE 802.11 standard, followed by the discussion of the most relevant literature concerning IEEE 802.11 energy efficiency mechanisms in mobile devices.

The main goal of mobile device energy management is to keep as long as possible the network interface in a low energy consumption state, usually called sleep mode. Unlike in awake mode, a mobile device in sleep mode cannot receive or transmit data. The most popular power saving mechanism for IEEE 802.11 network interfaces is the Power Save Mode (PSM) [1], usually referred to as Legacy-PSM. When operating in Legacy-PSM, all the transmitted data to a certain device in sleep mode is queued at the Access Point (AP). Later, after being notified via *Beacon* messages (usually sent every 100 ms) the device must wake-up to perform pending data polling (sending a *PS-Poll* message for each pending frame). As this mechanism is usually associated with higher delays [4], the last generation of mobile devices addressed the problem by implementing an adaptive mechanism to switch faster between awake and sleep modes, commonly named Adaptive-PSM. In Android, the Adaptive-PSM implementation switches between awake and sleep modes depending on the network traffic, allowing the IEEE 802.11 interface to stay awake only when there is traffic.

The Adaptive-PSM implementation in Android devices does not consider traffic type and importance when switching between awake and sleep modes, leading to several unnecessary switches to awake mode. Trying to overcome this limitation, Pyles et al. [5] proposed the Smart Adaptive PSM (SAPSM). SAPSM's main goal is to avoid that low priority applications switch the interface to awake, which results in energy savings for this type of applications, while the high priority ones still have good performance. The application priority is given based on the statistical information collected in the device using the proposed Application Priority Manager service. Although the authors argue that such approach might have benefits for non-technical users, it does not allow the application or the end-user to fully control the decision process. Furthermore, if a continuous media application is classified as high priority, the SAPSM mechanism will not be able to go back into sleep mode.

An extension to the common Android's Adaptive-PSM, aiming at improving the VoIP energy efficiency, was proposed by Pyles et al. The silence prediction based WiFi energy adaptation mechanism [6], SiFi, manages the device energy states according to the VoIP application characteristics. The proposed mechanism predicts the silence periods in the VoIP call and uses this information to

keep the wireless interface in sleep mode for a longer time. The results show 40% of energy savings, while keeping high voice quality. However, this approach is limited to VoIP applications, and the energy savings are closely related with the existence of silence periods.

A framework to reduce energy consumption of video streaming has been proposed by Csernai and Guly [7], aiming to dynamically adjust the awake interval according to the estimated video quality. The proposed mechanism was implemented in Android, but no accurate energy consumption study on the mobile equipment has been performed. An optimized power save algorithm for continuous media applications (OPAMA) was proposed by Bernardo et al. [8]. Although the OPAMA algorithm takes the end-user feedback into the process, it is limited by the AP configurations. The results show that OPAMA can save up to 44% of energy when compared with Legacy-PSM, but no real testbed validations have been performed. Korhonen and Wang [9] also proposed a mechanism to reduce energy consumption of multimedia streaming for UDP based applications. The mechanism dynamically adapts the burst intervals by analyzing network congestion. Despite of energy savings by sending the data into bursts, this proposal does not allow that the power control is driven by the end-users.

Ding et al. [10] introduce Percy, a mechanism that aims at reducing the energy consumption while keeping a low transfer delay for Web 2.0 flows. To achieve this goal a local proxy behind the AP was implemented. Energy savings are possible, since the device can go back into sleep mode when the proxy is queuing data to it. The assessment conducted in a mobile testbed shows energy savings between 44% and 67%. Nevertheless, as in other proxy-based approaches (e.g., [11]) the deployment of the solution is hard, since it requires changes in the access points. Additionally, these approaches do not consider end-users feedback.

To the best of our knowledge, this work advances the current state of the art by specifying an Android-based framework, which allows power saving mechanisms to be controlled by end-users. Such tool enables the possibility of including end-users and/or application preferences within the IEEE 802.11 interface power saving management.

3 EXPoSE: an Android framework for Extending Power Saving control to End-users

This section discusses the Android platform internal design in Section 3.1, followed by the presentation of the EXPoSE framework in Section 3.2.

3.1 Android overview and motivation

Android [3] is an open source operating system (OS), based on Linux, which is being widely used in mobile devices. As Android OS is mainly used in mobile devices, the battery management is a critical issue to be addressed. Recent studies [12] have shown that wireless interfaces of mobile devices represent a non-negligible part of the total energy consumed. Therefore, aiming at saving

energy, it is important to perform a proper management of the Android IEEE 802.11 interface. Figure 1 illustrates the Android IEEE 802.11 architecture.

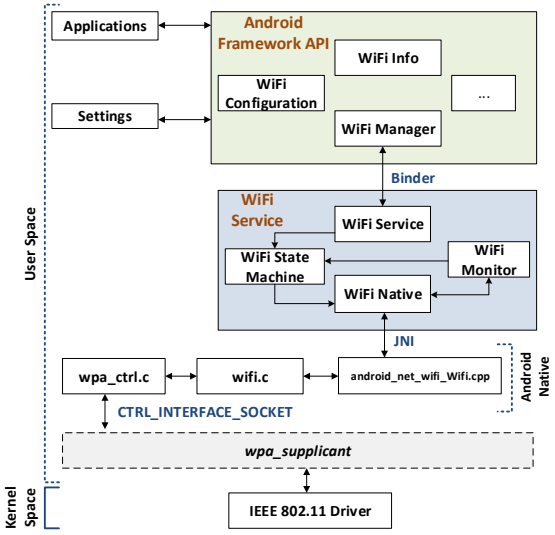


Fig. 1: Architecture of IEEE 802.11 in Android. (Based on [13])

To manage the configurations of the IEEE 802.11 interface (WiFi) applications must interact with the "*WiFi Manager*", which handles the communication with "*WiFi Service*". "*WiFi Service*" controls WiFi related communication between end-user and kernel spaces, being responsible for managing ("*WiFi State Machine*") and translating ("*WiFi Native*") all the requests. "*WiFi Native*" interfaces with the WiFi library, available as Android native code, through Java Native Interface (JNI). Finally, all the lower level calls to the IEEE 802.11 driver are performed through "*wpa_supplicant*".

Although the IEEE 802.11 architecture of Android is clear and well defined, it does not expose any IEEE 802.11 sleep related feature in the "Application Framework API" nor in the "WiFi Service". Therefore, the defined architecture does not allow the management of the IEEE 802.11 sleep functions at higher-layers (e.g., application). This paper addresses this issue by proposing enhancements to the current IEEE 802.11 architecture in Android, allowing power saving to be controlled by end-users, as described in the next section.

3.2 EXPoSE design

This section presents the Android framework for Extending Power Saving control to End-users (EXPoSE) design.

Figure 2 illustrates the architecture of the EXPoSE framework. The EXPoSE framework was implemented as an Android service (EXPoSE Service), plus a lower level control module included in the Android kernel. This module allows

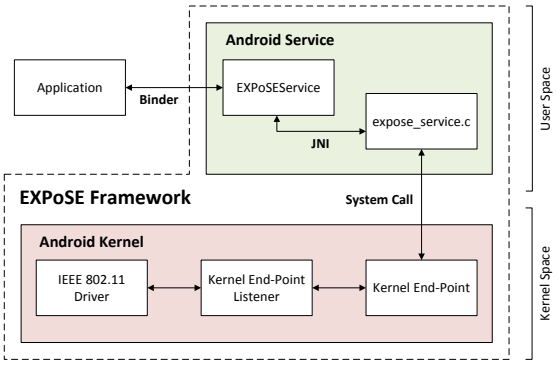


Fig. 2: EXPoSE framework architecture.

the IEEE 802.11 power saving functions to be exposed to higher level layers, enabling better control of power states.

To enable generic communication with the IEEE 802.11 driver, the developed Android kernel module is composed of two distinct components: the "Kernel End-Point" and the "Kernel End-Point Listener". The communication between the kernel end-point and the driver is performed through the proposed kernel end-point listener. Such abstraction plays an important role concerning energy efficiency, since, although the listener is always active, it is waiting in a semaphore and does not perform any additional processing (with extra energy costs).

Concerning the communication with the applications, the EXPoSE service can be configured in two distinct ways:

- **Pattern-based**: allows the application to configure the awake/sleep pattern over time. For instance, an application can specify that it must be awake for a certain period, α milliseconds, and it must be in sleep mode for a period of β milliseconds;
- **Maximum Allow Delay (MAD) definition**: enables the application to indicate that a maximum delay of γ milliseconds will be allowed. The control of the awake/sleep pattern over time will be performed by the EXPoSE service, taking into account the delay bound restriction.

To use the pattern-based approach, an application should indicate, at least, three distinct values. The first value is a flag to indicate if the specified pattern should be repeated over time. Such flag should be followed by two parameters, α and β, respectively, the awake and sleep periods in milliseconds. When using the MAD approach the application should only indicate a single value, γ, representing the maximum allowed delay in milliseconds.

As default Android Adaptive-PSM, the EXPoSE service also performs regular switching between sleep and awake modes. However, unlike Adaptive-PSM, the power modes switches are not based on the traffic load, but rather on application or end-users requirements. To change the IEEE 802.11 network interface to sleep mode for a certain period, EXPoSE changes the power mode on the IEEE 802.11

driver, forcing a NULL data frame with the Power Management flag enabled to be sent to the AP. Such action informs the AP that incoming data for that station should be queued, as in Legacy-PSM. Once the defined sleep period expires, the EXPoSE forces the interface to go back into awake mode and a NULL data frame to the AP with Power Management flag set to 0 is sent, allowing the queued data to be transmitted without any polling message.

The "EXPoSE service" interacts with the IEEE 802.11 driver through the "Kernel End-Point" by sending the time in milliseconds that the IEEE 802.11 network interface must be in sleep mode. Once the configured time expires, the "Kernel End-Point Listener" puts the interface back into awake mode. This scheme minimizes the interactions between user and kernel spaces, leading to a higher system level performance.

4 Experimental Evaluation

This section describes the experimental assessment performed in the testbed. Section 4.1 describes the evaluation goals, followed by the testbed presentation in Section 4.2. Finally, in Section 4.3, the results obtained in the testbed are presented and discussed.

4.1 Objectives

The experimental evaluation has two main goals. First, it aims to study the standard IEEE 802.11 power saving schemes, implemented in Android, in the presence of continuous media applications. The study includes the analysis of both energy efficiency and network-level performance metrics (e.g. delay) and the assessment of different application design options (e.g. packet size).

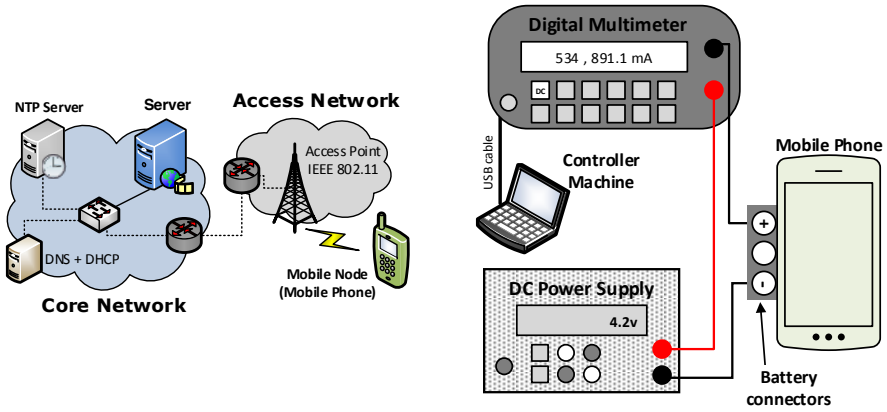
The second goal is to evaluate the EXPoSE approach effectiveness and to compare its performance with standard mechanisms.

4.2 Testbed setup

This subsection presents the IEEE 802.11 testbed and the energy measurement methodology to assess energy consumption of mobile phones.

Figure 3 illustrates the IEEE 802.11 and energy measurement testbed. Figure 3a depicts the IEEE 802.11 architecture, and the energy measurement components are detailed in Figure 3b.

The "Mobile Node" used in this setup was a LG P990 mobile phone, running Android 4.2.2 and the "Access Point" was a Cisco Linksys E4200. The machines in the *Core Network* ("Server", "NTP Server", and "DNS+DHCP" entities) were virtualized and run in a HP ProLiant DL320 G5p server. Besides the "Mobile Node", all the other machines were running Debian 7.0. All the traffic generated in the following tests has "Server" as source and "Mobile Node" as destination. Traffic generation was performed using the D-ITG version 2.8.1 [14].



(a) IEEE 802.11 architecture. (b) Energy measurement setup.

Fig. 3: IEEE 802.11 and energy measurement testbed

The mobile phone energy consumption assessment was addressed by extending a high-precision methodology [15] to support mobile devices. Figure 3b depicts the employed energy assessment testbed. The "Digital Multimeter" is a Rigol DM3061 unit, and supports up to 50.000 samples per second. This high-precision tool ensures the correct measurement of mobile phone energy consumption, since it will be possible to measure all the slight energy variations. The multimeter is managed by the "Controller Machine", which is a central control point for all the energy-related measurements. The communication between the "Control Machine" and the "Digital Multimeter" is performed using the Standard Commands for Programmable Instruments syntax (IEEE 488.2).

Rice et al. [16] were one of the first to explore the energy consumption in mobile phones. They proposed a methodology where a plastic battery holder replaces the battery, allowing the battery drop to be measured by using a high-precision measurement resistor in series between the holder cables and the battery. Although the accuracy behind this approach might be enough to measure mobile phone energy consumption, it still depends on the battery discharging pattern. Therefore, in the methodology used in this paper, an external "DC Power Supply" was employed. Nevertheless, as the mobile phone is expecting to receive battery status information via the smart battery system, the external power supply can not be directly used. The solution used in the measurements presented in this paper was to employ a specific voltage to allow the RT9524 unit of LG P990 (unit that controls the phone charging process) to be changed to "Factory Mode". This mode allows to supply the system using an external power supply and without connecting a battery.

The energy consumption of the IEEE 802.11 interface described in the next subsections is given by the difference between the mobile phone total energy consumption and the baseline energy consumption with the device operating in airplane mode (all radios off) with the display brightness at 100%. Each

performed test has a duration of 60 seconds, and all the results presented include 20 distinct runs using with a confidence interval of 95%.

4.3 Results

This section discusses the obtained results regarding the No-PSM, Legacy-PSM and Adaptive-PSM performance when receiving data from a continuous media application, compared with the EXPoSE approach.

Impact of packet size: This section discusses the impact of the packet size in energy consumption of the three power saving approaches in study, namely No-PSM, Legacy-PSM and Adaptive-PSM. The data was sent with a fixed transmission rate of 100 packets per second. The packet size ranges from 200 to 1400 bytes. Figure 4 depicts the total energy consumed by the IEEE 802.11 interface (in Joule) during 60 s by each power saving algorithm, according to the employed packet size in bytes (x-axis).

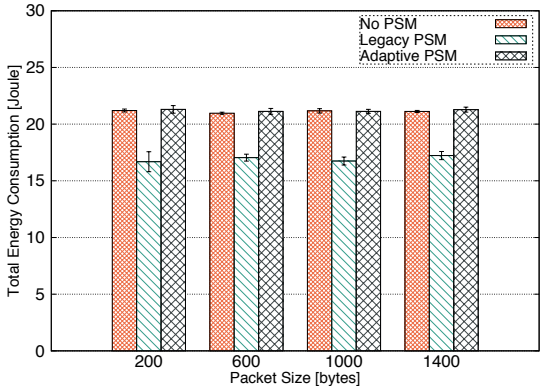
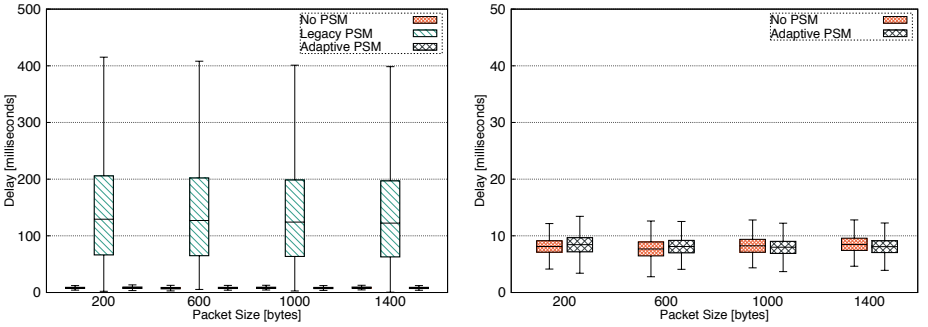


Fig. 4: Total energy consumed by the IEEE 802.11 interface with different packet sizes.

The obtained results depict the Adaptive-PSM limitations in the presence of continuous media applications. Since in these applications there will be always data being transmitted from the core network to the mobile phone, the Adaptive-PSM algorithm does not have enough opportunities to sleep. Furthermore, due to the energy costs of multiple transitions between awake and sleep modes, this dynamic approach might consume more energy than No-PSM. When employing the Legacy-PSM the energy savings are between 18% and 21% compared to No-PSM and Adaptive-PSM schemes, respectively.

Concerning the relationship between packet size and energy consumption, it is possible to see that the packet size has a negligible impact on the total energy consumed. Such results highlight the energy benefits of using larger packets, since the energy cost per transmitted byte will be much lower.

Besides the energy consumption behavior, the impact of power saving algorithms on application performance should also be considered. Figure 5 shows a *boxplot* representing the one way delay, in milliseconds, for all the packets transmitted using the different algorithms. Figure 5a depicts the delay for all the algorithms, while the Figure 5b zooms the same data only for No-PSM and Adaptive-PSM schemes.



(a) One way delay for all algorithms

(b) One way delay for No-PSM and Adaptive-PSM

Fig. 5: One way delay for different packet sizes.

Legacy-PSM introduces considerably more delay, when compared to No-PSM and Adaptive-PSM. The mean delay (second quartile) obtained for the Legacy-PSM is around 125 ms, while for No-PSM and Adaptive-PSM the mean delay is between 7 and 9 ms. Additionally, the No-PSM and Adaptive-PSM maximum delay never exceeds 14 ms and the Legacy-PSM has delays up to 400 ms. Again, the packet size does not reveal any impact on the results.

These results show that the energy savings (between 18% and 21%) obtained with the Legacy-PSM do not establish a good energy / performance trade-off, since there is a high impact on the application delay. Due to the polling phase, Legacy-PSM also generates packet loss, but always lower than 0.2%. Furthermore, the impact of packet size on the energy consumption is almost absent. Concerning the application design, the depicted data showed that using larger packets is highly preferable to improve overall energy consumption.

Impact of transmission rate: This section investigates the impact of the transmission rate on total energy consumption of No-PSM, Legacy-PSM and Adaptive-PSM. In this evaluation the packet size was fixed to 1000 bytes, with the transmission rate varying between 50 and 250 packets per second. Figure 6 presents the total energy consumed by the IEEE 802.11 interface (in Joules) during 60 s with the different transmission rates (x-axis). The results show that in both No-PSM and Adaptive-PSM it is possible to establish a linear relationship between the energy consumption over time and the transmission rate. When using Legacy-PSM, the energy consumption also increases with transmission rates,

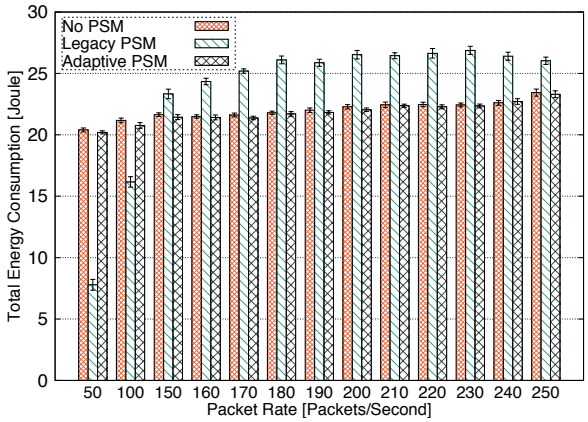
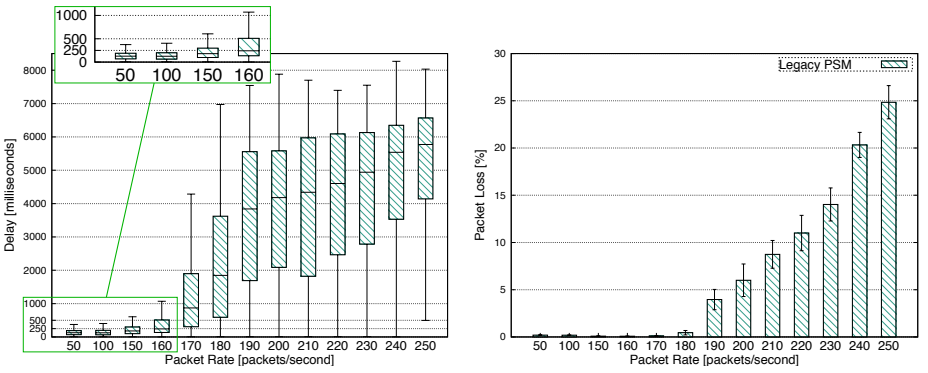


Fig. 6: Total energy consumed by the IEEE 802.11 interface with distinct transmission rates.

but only for rates up to 180 packets per second. With transmission rates above 180 packets per second, Legacy-PSM energy consumption is almost constant. Furthermore, Legacy-PSM only outperforms No-PSM and Adaptive-PSM for the lowest studied transmission rates.

Thus, in order to properly investigate the Legacy-PSM behavior, the one way delay and packet loss metrics were analyzed. No packet loss was detected with the No-PSM and the Adaptive-PSM schemes, and the mean delay ranges between 7 and 9 ms. The maximum delay observed was always lower than 14 ms. The *boxplot* depicting the delay and the packet loss rate for the Legacy-PSM are illustrated, respectively, in Figures 7a and 7b.



(a) One way delay (b) Packet loss rate

Fig. 7: One way delay and loss rate for Legacy-PSM with distinct transmission rates.

The Quality of Service (QoS) attained using Legacy-PSM is strongly affected by the transmission rate, as depicted by the mean delay for rates greater or equal

than 160 packets per second. Apart from the unacceptable delay, Legacy-PSM also affects application performance by introducing a non negligible packet loss for rates above or equal to 190 packets per second. Such behavior is related to the Legacy-PSM protocol design, where the device must send one *PS-Poll* to the access point to request each pending frame [4]. The long delays resulting from the protocol polling mechanism also explain the depicted packet loss, since various packets are dropped in the access point queues due to time constraint violation.

In short, when analyzing the behavior of Legacy-PSM and Adaptive-PSM it is possible to conclude that they are not able to establish a proper energy / performance trade-off for continuous media applications. Legacy-PSM strongly affects the application performance, whereas the Adaptive-PSM can keep the application requirements, but without achieving significant energy savings.

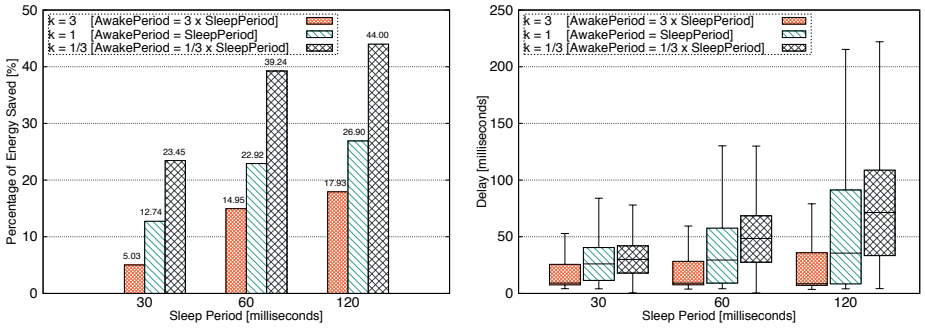
EXPoSE pattern-based sleep approach: This section explores the employment of EXPoSE using the pattern-based sleep approach configured by the application, as described in Section 3.2. All the results presented next were performed using a fixed packet size of 1000 bytes with a constant transmission rate of 200 packets per second. This configuration was selected, since it represents a scenario where the Legacy-PSM performance is already worse than both No-PSM and Adaptive-PSM (see Figure 6).

As the goal of this assessment is to study the EXPoSE impact on the energy consumption and network performance, 9 distinct sleep patterns were selected as illustrated in Table 1. Apart from the parameters required to configure the EXPoSE pattern-based solution, the table also depicts a constant, κ, associated with each test. This constant allows to establish a relationship between the configured periods, that means $AwakePeriod = \kappa \times SleepPeriod$.

Table 1: EXPoSE pattern-based configurations

Test ID	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**
Loop Flag	1	1	1	1	1	1	1	1	1
SleepPeriod (ms)	30	30	30	60	60	60	120	120	120
AwakePeriod (ms)	90	30	10	120	60	20	360	120	40
κ	3	1	1/3	3	1	1/3	3	1	1/3

Figure 8a shows the EXPoSE pattern-based solution energy savings compared to Adaptive-PSM for the different configurations. As expected, the results show a direct relationship between the energy savings and the total time in sleep mode. Even for the scenarios with κ=3, where the time in sleep mode is 25% of the total time, the energy savings are up to 17.93% for the scenario with a sleep period equal to 120 ms. When reducing the awake period, an improvement in energy savings can be observed. With κ=1, where the awake and sleep periods are equal, the savings are up to 26.90%. If the awake period is reduced to 1/3 of the sleep period (i.e., κ=1/3) energy savings are 23.45%, 39.24% and 44.00% for configured sleep periods of 30, 60 and 120 ms, respectively.



(a) Energy savings compared with Adaptive-PSM

(b) One way delay

Fig. 8: Energy savings and one way delay for EXPoSE pattern-based approach.

The delay results, depicted in Figure 8b, show that EXPoSE impact on the delay is not negligible, such as with for Adaptive-PSM. For a similar scenario (with a rate of 200 packets per second and using packet size of 1000 bytes) the Legacy-PSM mean delay is around 4 s, with a maximum delay higher than 7 s. Moreover, packet loss with EXPoSE pattern-based approach was always lower than 0.02%, against 6.00% with Legacy-PSM.

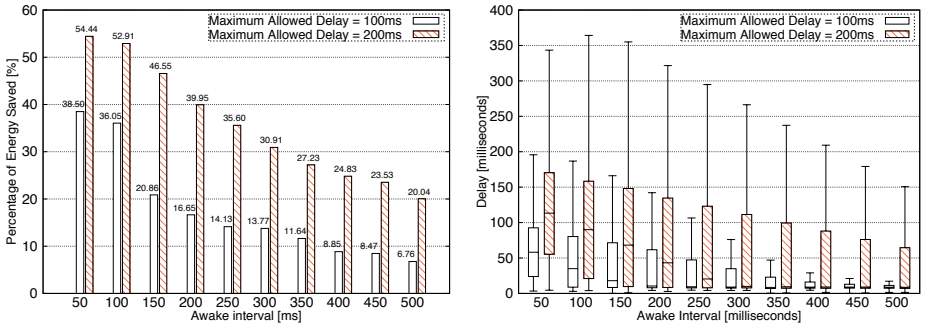
In the studied scenarios, the EXPoSE pattern-based approach shows mean delays bellow 100 ms, enabling the possibility to be employed with continuous media application, as for instance, video streaming.

EXPoSE maximum allowed delay approach: This section studies the EXPoSE maximum allowed delay approach. In this evaluation, two applications were selected: one with a maximum allowed delay equal to 100 ms, and another allowing delays up to 200 ms. Both applications were emulated using a transmission rate of 200 packets per second, with packets of 1000 bytes length.

The main objective of EXPoSE's maximum allowed delay approach is to keep the application QoS requirements within specified bounds. Therefore, this section investigates the minimum awake period required to not exceed the defined maximum allowed delay. As defined in the EXPoSE maximum allowed delay mechanism, the sleep period will be equal to the configured maximum allowed delay.

Figure 9a shows the energy savings percentage compared with the Adaptive-PSM algorithm on the y-axis, while the x-axis depicts the tested awake periods, ranging from 50 ms to 500 ms. The one way delay for the same assessment is illustrated in Figure 9b.

The results show that to keep the application delay within the defined bounds, the awake period should be defined as 300 ms and 450 ms for maximum allowed delays of 100 ms and 200 ms, respectively. Energy savings for the lowest maxi-



(a) Energy savings compared with Adaptive-PSM

(b) One way delay

Fig. 9: Energy savings and delay for EXPoSE maximum allowed delay scenarios.

mum allowed delay are 13.77% and 23.53% when the application supports delays up to 200 ms, respectively. Nonetheless, if the application allows that only 75% of the packets (*boxplot* third quartile) arrive within the configured limits, it is enough to be awake during 50 ms and the energy savings for 100 ms and 200 ms maximum allowed delay are 38.50% and 54.44%, respectively, compared to Adaptive-PSM.

5 Conclusions

The fast growth of mobile devices deployment created new demands concerning the energy efficiency of the IEEE 802.11 network interfaces, which is one of the core access technologies supporting the communication of those devices. As Android is one of the most used platforms in portable equipment, the IEEE 802.11 power saving mechanisms in this system should be scrutinized.

Besides investigating the performance of IEEE 802.11 power saving techniques available in Android, this paper proposes an Android framework for Extending Power Saving control to End-users (EXPoSE), which enables an end-user and application based control of the IEEE 802.11 network interface power management. The achieved results showed that EXPoSE approaches, namely the pattern-based and the maximum allowed delay, are more energy efficient than both Legacy-PSM and Adaptive-PSM schemes. Depending on the scenarios and applications requirements, the EXPoSE energy savings, compared to Adaptive-PSM, can go up to 23.53% without violating the application delay constraints. Moreover, if some additional delay is acceptable (e.g., only 75% of the packets arriving on time), the energy savings can be more than 50%, compared to Adaptive-PSM.

Furthermore, the obtained results depicted the EXPoSE capabilities to improve continuous media applications energy efficiency, which is not well supported by Legacy-PSM and Adaptive-PSM strategies.

Acknowledgments

This work was partially supported by the COST Action IC0906, as well as by the iCIS project (CENTRO-07-ST24-FEDER-002003), co-financed by QREN, in the scope of the Mais Centro Program and European Union's FEDER. The first author was also supported by the Portuguese National Foundation for Science and Technology (FCT) through a Doctoral Grant (SFRH/BD/66181/2009).

References

1. IEEE: IEEE std 802.11-2012 (revision of ieee std 802.11-2007). (2012) 1–2793
2. Costa-Pérez, X., Festag, A., Kolbe, H.J., Quittek, J., Schmid, S., Stiemerling, M., Swetina, J., van der Veen, H.: Latest trends in telecommunication standards. SIGCOMM Comput. Commun. Rev. **43**(2) (April 2013) 64–71
3. Android Open Source Project: Android open-soruce software stack (2014)
4. Tsao, S.L., Huang, C.H.: A survey of energy efficient MAC protocols for IEEE 802.11 {WLAN}. Computer Communications **34**(1) (2011) 54 – 67
5. Pyles, A.J., Qi, X., Zhou, G., Keally, M., Liu, X.: SAPSM: smart adaptive 802.11 PSM for smartphones. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing. UbiComp '12, New York, NY, USA, ACM (2012) 1120
6. Pyles, A.J., Ren, Z., Zhou, G., Liu, X.: Sifi: Exploiting voip silence for wifi energy savings insmart phones. In: Proceedings of the 13th International Conference on Ubiquitous Computing. UbiComp '11, New York, NY, USA, ACM (2011) 325–334
7. Csernai, M., Gulyas, A.: Wireless adapter sleep scheduling based on video qoe: How to improve battery life when watching streaming video? In: ICCCN 2011. (July 2011) 1–6
8. Bernardo, V., Curado, M., Braun, T.: An IEEE 802.11 energy efficient mechanism for continuous media applications. Sustainable Computing: Informatics and Systems **4**(2) (2014) 106–117
9. Korhonen, J., Wang, Y.: Power-efficient streaming for mobile terminals. In: Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video. NOSSDAV '05, ACM (2005) 39–44
10. Ding, N., Pathak, A., Koutsonikolas, D., Shepard, C., Hu, Y., Zhong, L.: Realizing the full potential of psm using proxying. In: INFOCOM, 2012 Proceedings IEEE. (March 2012) 2821–2825
11. Dogar, F.R., Steenkiste, P., Papagiannaki, K.: Catnap: Exploiting high bandwidth wireless interfaces to save energy for mobile devices. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services. MobiSys '10, New York, NY, USA, ACM (2010) 107–122
12. Cui, Y., Ma, X., Wang, H., Stojmenovic, I., Liu, J.: A survey of energy efficient wireless transmission and modeling in mobile cloud computing. Mobile Networks and Applications **18**(1) (2013) 148–155
13. Amuhong: IEEE 802.11 architecture in Android. (2014)
14. Botta, A., Dainotti, A., Pescapè, A.: A tool for the generation of realistic network workload for emerging networking scenarios. Computer Networks **56**(15) (2012)
15. Bernardo, V., Curado, M., Staub, T., Braun, T.: Towards energy consumption measurement in a cloud computing wireless testbed. In: International Symposium on Network Cloud Computing and Applications (NCCA). (2011) 91–98
16. Rice, A., Hay, S.: Measuring mobile phone energy consumption for 802.11 wireless networking. Pervasive Mob. Comput. **6**(6) (December 2010) 593606

Appendix B

This appendix introduces the Android OS. First, an overview is given, followed by the Android architecture. Next, the implementation of the IEEE 802.11 in Android is presented. Finally, a discussion about the open issues and limitations is provided.

Overview

The Android story dates back to 2002 when the two co-founders of Google, Larry Page and Sergey Brin, attended a talk [Yaghmour, 2013]. The talk was about the Sidekick phone, one of the first multi function phones with internet access, produced by Danger Inc. The Sidekick phone had the Google search engine as default and both Larry and Sergey started to use this device. However, this phone was not a commercial success and, therefore, the Danger's CEO, Andry Rubin decided to discontinue the Sidekick device. Some time later, Andry Rubin wanted to create an open source OS for mobile phones and the Android Inc was created. The Android Inc was then acquired by Google in August of 2005.

In November 2007, the Open Handset Alliance (OHA) announced the Android Inc acquisition by Google. In September 2008, a first version of Android (1.0) was released.

Now, the Android OS is in the 4.4 version, which has evolved over the years, with the release of several versions. Moreover, the Android OS just exceeded the 1 billion activated devices.

Apart of the Android OS evolution in the last years, when talking about Android OS one of the most frequent words presented is Dalvik. In fact, the Dalvik is an important component of the Android OS. The Dalvik is a virtual machine, like Java virtual machine, but it was specially developed for the Android mobile phones requirements, i.e. consume less CPU and memory. It is the core of any Android application, since the Android byte code is interpreted in Dalvik. The Android applications byte code are

different from Java ones. The Android byte code is stored in a special format called Dex, which has some optimizations, in order to minimize the memory usage [Yaghmour, 2013].

The Android was designed to meet security and performance requirements. Therefore, each Android application runs in a separated process with a different instance of Dalvik. In order to instantiate these processes with high performance, a special process called Zygote was developed. The Zygote name was used in a parallelism to the first cell of an organism. In fact, the Zygote process is responsible to create multiple Dalvik instances. It starts at boot and preloads a Dalvik instance and the applications core libraries. When an application starts, the Zygote process is forked to create another process which runs the application. This allows the Dalvik instances to be created very fast. However, since each application runs in a different process this could be a problem concerning the memory. The memory is reduced by sharing all the possible libraries, needed by applications, between the Dalvik instances.

More recently, with the release of Android KitKat 4.4, a new experimental Android Virtual machine was made available. The new Android run time (ART) differs from Dalvik by changing the concept of just in time (JIT) introduced in Android version 2.2. The JIT is a component that translates and analyses the applications code to run faster. When an application is downloaded it stays in the device storage and, when is launched for the first time, it is compiled and placed in Random Access Memory (RAM). This leads to an overhead when the applications are firstly loaded. However, after being in RAM the load of applications is very fast. A problem is inherent to this concept, since, when an application is removed from the RAM, the whole process starts over and the applications outside RAM will be heavy to load. The ART uses another concept, the ahead of time (AOL). As opposed to JIT, the AOL pre-compiles the applications automatically when they are downloaded. Consequently, the applications will use more space in the device. However, the load and execution of applications is quite faster. This could lead to a lower CPU utilization and, therefore, to an improvement of battery life.

Android Architecture

Figure 1 illustrates the Android architecture. The architecture is composed by six main layers namely:

- **Application layer**: The layer that contains the applications running in an Android device, such as the home, contacts, browser and others;

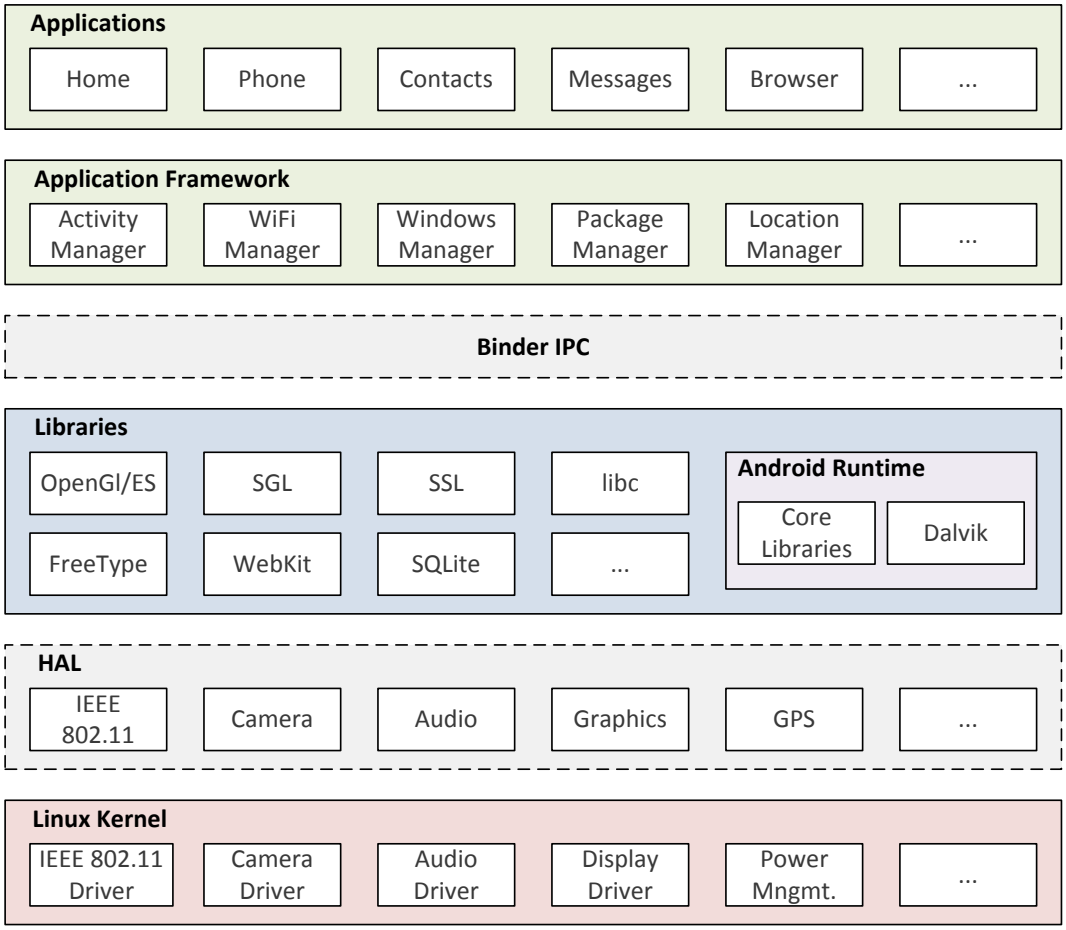


Figure 1: Android architecture. Based on [Android, 2014b] [Android, 2014c]

- **Application framework layer**: The application framework layer contains the frameworks that can be accessed by applications through an API;
- **Binder Inter Process Communication (IPC) layer**: This is a mechanism that allows the application framework layer to communicate with the libraries layer. The binder is provided at the framework level, however, all the communication is hidden, i.e. the communication is transparent for the developer;
- **Libraries layer**: The libraries layer is divided in two components, the Android runtime and the Android libraries. The Android runtime contains the core libraries and the dalvik virtual machine. The Android libraries contain all the non core

libraries. Almost every functionality that could be used by the framework APIs, communicates with one service. This allows the access to the hardware, with the appropriate restrictions, in the application level;

- **Hardware Abstraction Layer (HAL)**: The Android system communication with the hardware relies on the HAL. The HAL is a standard interface, normally provided by the manufactures, which aims the communication between the system and hardware based on provided modules. As opposed to direct communication with the driver.

- **Kernel layer**: The Android uses a linux kernel with some important additions for embedded platforms. It uses, apart from others, an aggressive memory management system, which preserves the memory, wakelocks and a driver for inter process communication.

IEEE 802.11 Implementation on Android

Figure 2 illustrates the IEEE 802.11 implementation architecture based on Android 4.0. The architecture components are mostly implemented in the user space and communicate with the IEEE 802.11 driver presented in the kernel space.

The user space is divided in four parts, the Android applications, the Android framework API, provided by the Android SDK, the Android services and the Android native code.

The Android SDK is based in the Java language and provides an API that allows the applications to communicate with the system services, through a Binder. The Binder, as explained before, is a specific inter process mechanism for communication and also a remote method invocation system. The android SDK is the level at which application developers should be concerned about.

The system services are daemons, i.e. processes running in background. The services receive messages, from the applications, through the provided APIs and call the correspondent native methods through JNI. The native methods are mostly written in C and C++.

When an Android application or the Android OS settings intend to manage something related to the WiFi, must firstly communicate with the "*Wifi Manager*". Then, the "*Wifi Manager*" sends a message to the "*Wifi Service*" through a Binder.

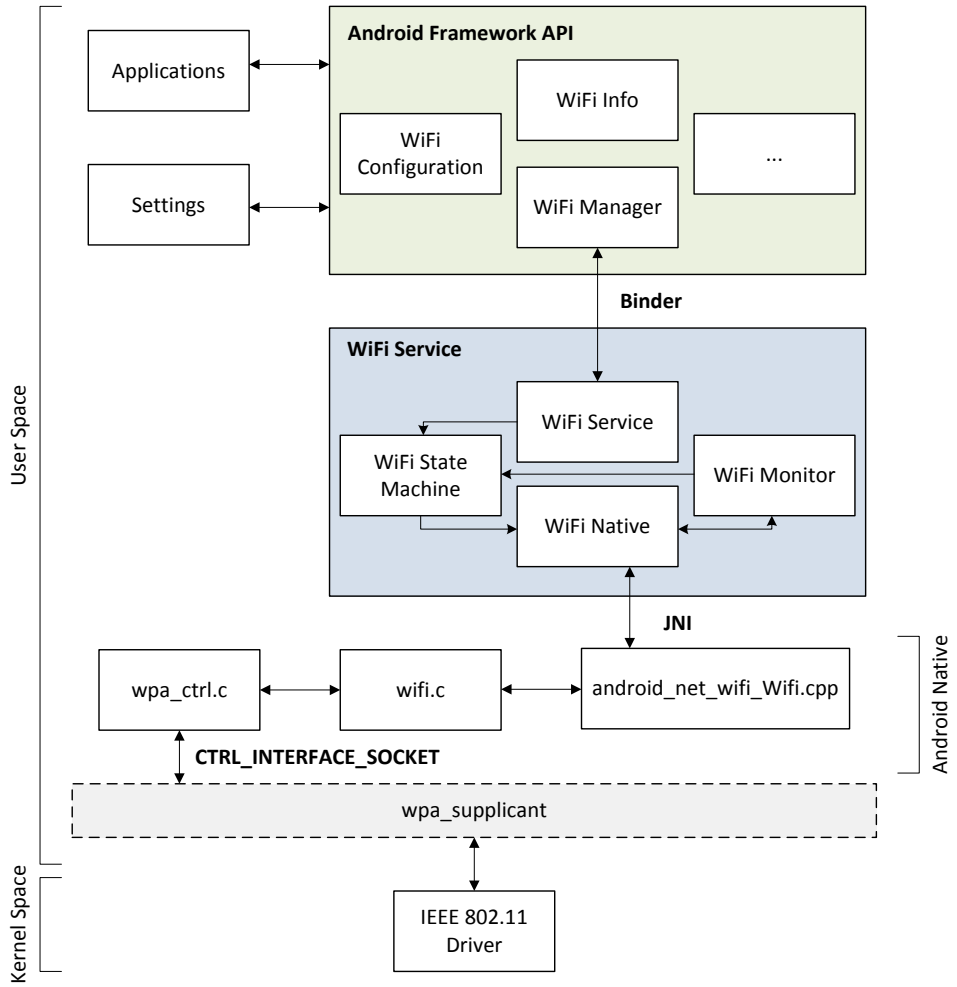


Figure 2: IEEE 802.11 implementation architecture in Android. Based on [Amuhong, 2014]

In WiFi Services, when the "*Wifi Service*" receives a message from the "*Wifi Manager*", a command is sent to "*Wifi State Machine*". The "*Wifi State Machine*" initializes the "*Wifi Monitor*" that tries to connect to the "*wpa_supplicant*" through the "*Wifi Native*". When the connections is established, the "*Wifi Monitor*" informs the "*Wifi State Machine*" that is possible to send commands to the "*wpa_supplicant*". Moreover, the "*Wifi Monitor*" listens to the events that occurs in the "*Wifi Native*" to communicate them to the "*Wifi State Machine*".

The "*Wifi Native*" is the class responsible to communicate with the low level layer through JNI which implements the methods called in the java class. Therefore, these

methods will communicate through a socket with the "*wpa_supplicant*" that sends commands to the IEEE 802.11 Driver in order to communicate with the hardware.

Summary

In this chapter a brief introduction to the Android history was presented. Moreover, in a first overview, one of the most important components of the Android OS, the Dalvik, was discussed and compared with the new ART. As showed, the ART is the future of the Android virtual machine, which could lead to significant improvements in the Android devices performance, by reducing the CPU load and memory. Additionally, this new virtual machine could also improve the battery lifetime of the Android devices.

The Android stack architecture was presented and its main layers were described. The Android architecture has a well defined layered structured, with a specific purpose for each layer, as well as layer interactions. The application and framework layers are the layers which are usually dominated by application developers. However, to take full advantage of Android it is necessary to know and understand the lower layers.

Since this work is based on IEEE 802.11, one of the first goals was to verify its implementation in Android. This allows to understand all the interactions between the layers and components used by IEEE 802.11 and, therefore, its study and modification. This would not be possible if the Android OS was not open source. Therefore, the Android OS is a very important tool to validate and test the implementation of the proposed mechanisms in real devices.

Appendix C

This appendix presents the results obtained in the first semester, concerning the transmission of UDP traffic that modulates an on/off traffic pattern.

The tests performed have no extra applications running in background and the tests concerning the energy consumed by IEEE 802.11 technology have the display brightness at 100%. Each test has a duration of 60 seconds and was done with a rate of 1000 samples per second. All the results are based on 10 runs and are exhibited in the charts with a confidence interval of 95% assuming that the measurements follow a normal distribution.

On/Off UDP traffic scenario

In this scenario the energy consumed by the IEEE 802.11 power management modes, No-PSM and Adaptive-PSM, will be analyzed while the device receives on/off UDP traffic.

For each IEEE 802.11 power management mode, two tests were performed with the application sending rate at 8 Mbps. The 8 Mbps. Both tests in this scenario were performed by sending traffic only in some intervals during the total energy assessment time. The used intervals to send the traffic were set to 10 and 20 seconds. This allows the understanding of the No-PSM and Adaptive-PSM modes behavior when the traffic is interrupted for a given time, since in a normal usage of a device, the network traffic is not always presented.

Figure 3 illustrates the performed tests, which sends on/off UDP traffic with 20 seconds of interval. Therefore, the traffic is sent between the 10 and 30 seconds, followed by a 20 seconds period where no traffic is sent. Finally, between the 50 and 70 seconds, the traffic is sent again.

In the test with 10 seconds of interval, the traffic is started in the following instants: 10, 30 and 50 seconds.

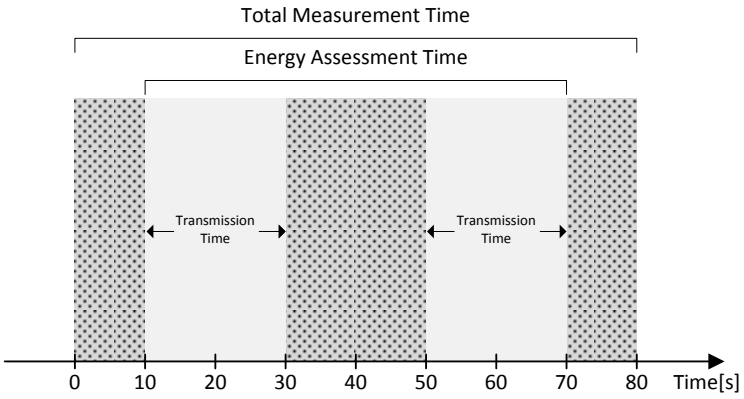


Figure 3: On/off UDP traffic with intervals of 20 seconds.

Analysis

Figure 4 presents the energy consumption results of the IEEE 802.11 power management modes in the presence of on/off UDP traffic. The y-axis depicts the total energy consumption of the smartphone (in Joule), while x-axis represents the interval in seconds between the sent data traffic.

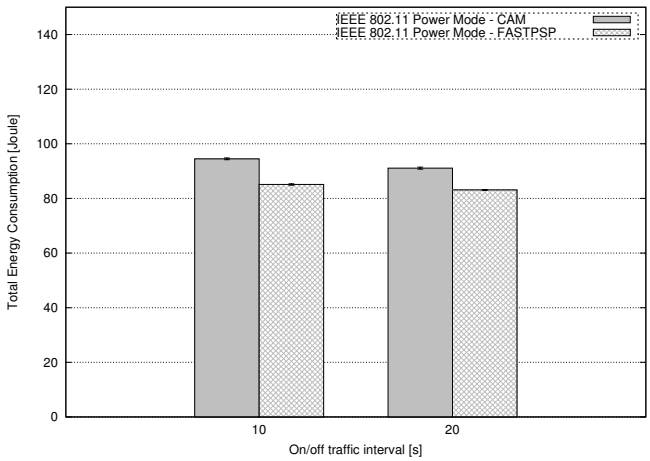


Figure 4: Total energy consumed in IEEE 802.11 power management modes with on/off UDP traffic.

As expected, the Adaptive-PSM mode has the best performance, concerning energy consumption, when compared with the No-PSM mode. This happens because the traffic is not sent in the total amount of the test. Therefore, in the intervals between the sent

traffic, the Adaptive-PSM mode is able to change the wireless interface to sleep, saving energy.

Figures 5 and 6 represent the power consumed by smartphone (y-axis), over the time (x-axis), while receiving network traffic in intervals of 10 and 20 seconds. Both figures showed that while theNo-PSM mode always remain at a high power level, the Adaptive-PSM mode switches between a high power level and a low level of power depending on the traffic presented in the network. This leads to a lower consumption of the Adaptive-PSM mode when compared with the No-PSM mode.

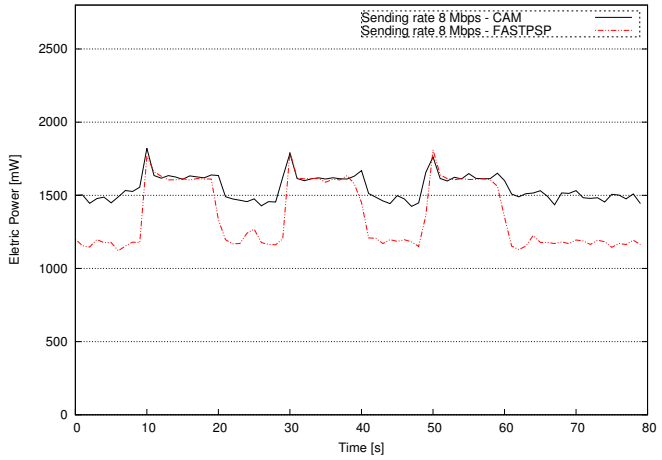


Figure 5: Power consumed in IEEE 802.11 modes with on/off traffic sent in 10 s intervals.

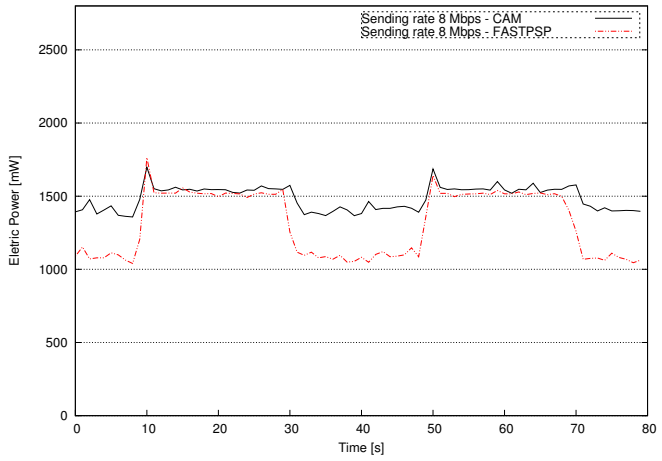


Figure 6: Power consumed in IEEE 802.11 power management modes with on/off UDP traffic sent in 20 seconds intervals, along the time.

Furthermore, it is noticeable that to change the power mode, the Adaptive-PSM incurs with extra energy costs, as shown by the energy transactions between the sleep state and the constantly awake state.