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.

source: https://doi.org/10.7892/boris.53977 | downloaded: 28.3.2024

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.

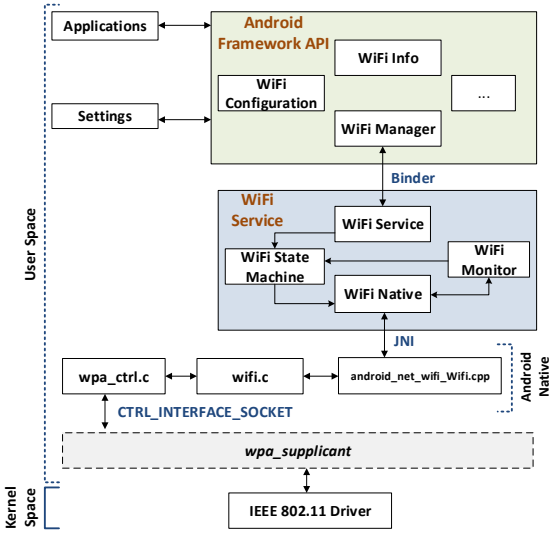


Fig. 1: Architecture of IEEE 802.11 in Android. (Based on [13])

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

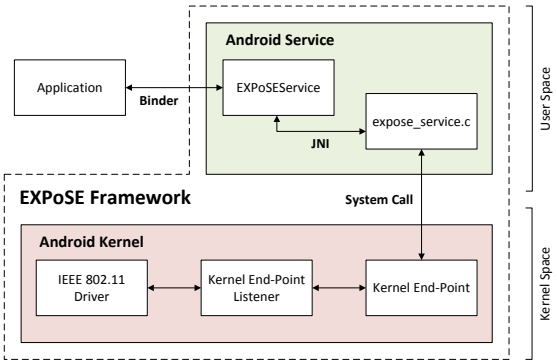


Fig. 2: EXPoSE framework architecture.

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).

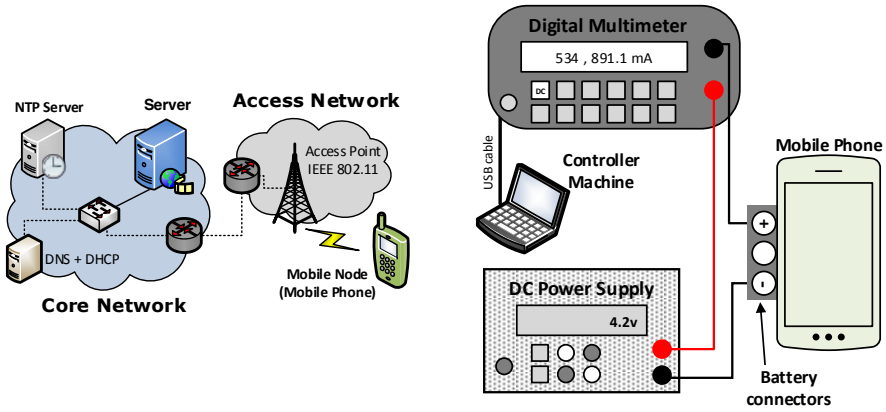
The second goal is to evaluate the EXPoSE approach effectiveness and to compare its performance with standard mechanisms.

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).

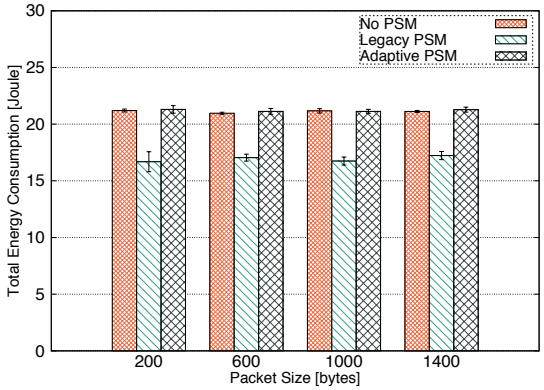
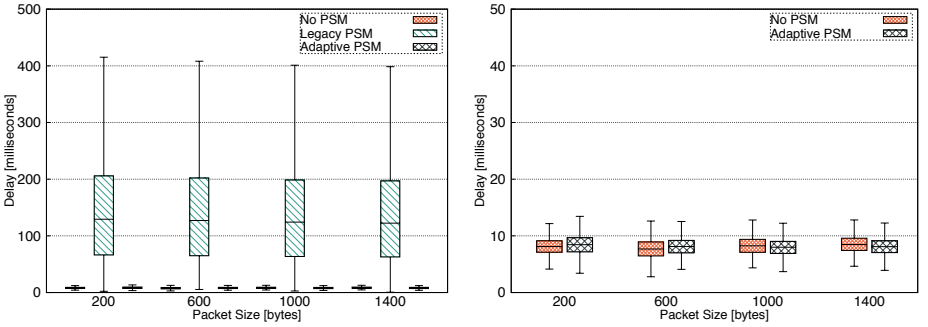


Fig. 4: Total energy consumed by the IEEE 802.11 interface with different packet sizes.

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,

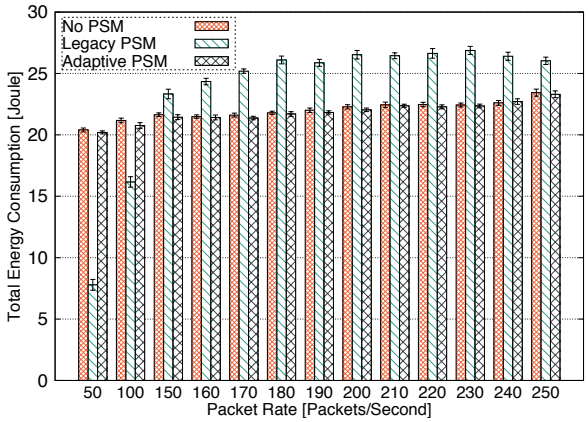
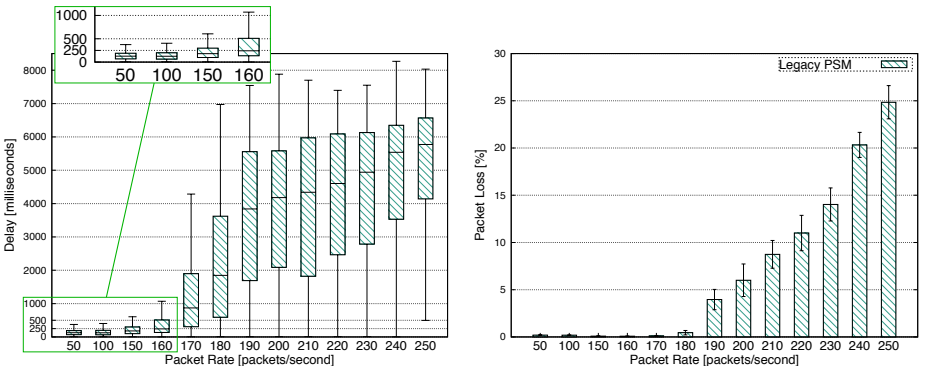


Fig. 6: Total energy consumed by the IEEE 802.11 interface with distinct transmission rates.

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.

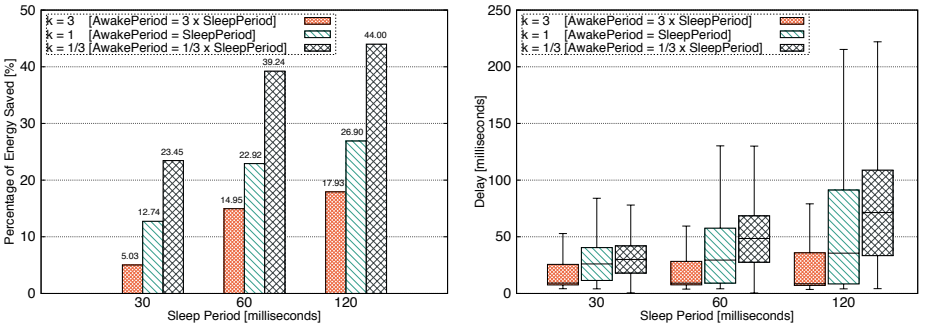
EXPoSE pattern-based sleep approach: This section explores the employment of EXPoSE using the pattern-based sleep approach configured by the application, as described in Section 3.2. All the results presented next were performed using a fixed packet size of 1000 bytes with a constant transmission rate 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.

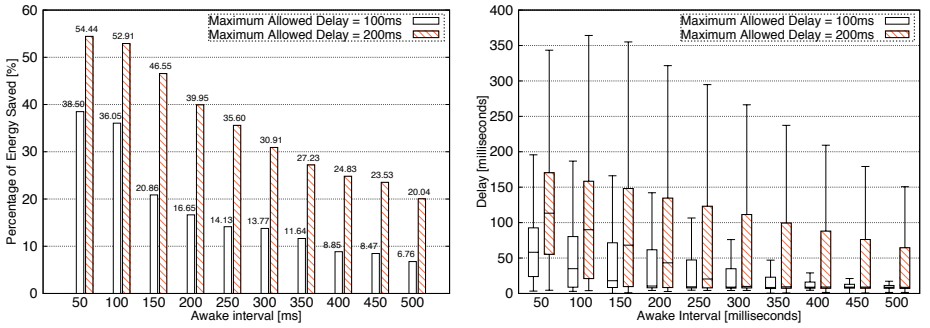
In the studied scenarios, the EXPoSE pattern-based approach shows mean delays bellow 100 ms, enabling the possibility to be employed with continuous media application, as for instance, video streaming.

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