802.11 Power-Saving Mode for Mobile Computing in Wi-Fi hotspots: Limitations, Enhancements and Open Issues

G. Anastasi[a], M. Conti[b], E. Gregori[b], A. Passarella[b,*]

Pervasive Computing & Networking Laboratory (PerLab)

[a]Dept. of Information Engineering, University of Pisa
Via Diotisalvi 2 - 56122 Pisa, Italy
g.anastasi@iet.unipi.it

[b]CNR - IIT Institute
Via G. Moruzzi, 1 - 56124 Pisa, Italy
{marco.conti, enrico.gregori, andrea.passarella}@iit.cnr.it

Abstract. Nowadays Wi-Fi is the most mature technology for wireless-Internet access. Despite the large (and ever increasing) diffusion of Wi-Fi hotspots, energy limitations of mobile devices are still an issue. To deal with this, the standard 802.11 includes a Power-Saving Mode (PSM), but not much attention has been devoted by the research community to understand its performance in depth. We think that this paper contributes to fill the gap. We focus on a typical Wi-Fi hotspot scenario, and assess the dependence of the PSM behavior on several key parameters such as the packet loss probability, the Round Trip Time, the number of users within the hotspot. We show that during traffic *bursts* PSM is able to save up to 90% of the energy spent when no energy management is used, and introduces a limited additional delay. Unfortunately, in the case of long inactivity periods between bursts, PSM is not the optimal solution for energy management. We thus propose a very simple Cross-Layer Energy Manager (XEM) that dynamically tunes its energy-saving strategy depending on the application behavior and key network parameters. XEM does not require any modification to the applications or to the 802.11 standard, and can thus be easily integrated in current Wi-Fi devices. Depending on the network traffic pattern, XEM reduces the energy consumption of an additional $20 - 96\%$ with respect to the standard PSM.

Keywords: 802.11, Wi-Fi, Power-Saving Mode, Network Architecture & Design, Mobile Computing, Network Protocols, Performance of Systems

1 Introduction

Since the introduction of the 802.11 standard in 1997, 802.11 wireless LANs (also known as Wi-Fi hotspots) have become more and more popular. Installations of Wi-Fi hotspots are nowadays very frequent, for example in company and education buildings, coffee shops, airports, and so on. Figure 1 shows a simple Wi-Fi installation, where users carrying *mobile hosts* (e.g., laptops, PDAs, ...) exploit an *Access Point* to connect to legacy Internet services. This is the scenario used in the paper.

Despite its increasing popularity, Wi-Fi still presents several problems that are far to be solved. One of the most important is the energy consumption of 802.11 wireless interfaces. Wireless cards have shown to account for about 10% of the total energy consumption in current laptops [1, 7]. This percentage grows up to 50% in hand-held devices [1, 31], and even beyond in smaller form-factor prototypes [41]. Even worse, the difference between battery capacities and the requirements of electronic components is expected to increase in the near future [40]. Energy management is hence a core enabling factor for the Wi-Fi technology[1].

*This work has been carried out while A. Passarella was with the Department of Information Engineering of the University of Pisa.

[1]In this paper we talk about "energy management" instead of "power management", though the latter keyword is quite more diffused in the literature. Actually, this paper is not about optimizing the *power* consumption of 802.11, i.e., by adjusting the transmission power or the receiver sensitiveness. Rather, it is about optimizing the *time intervals* spent by the wireless interface in the different 802.11 power modes, in order to minimize the *energy* consumed to perform networking activities.

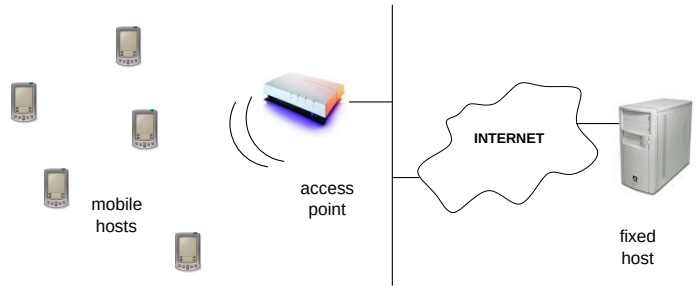


Figure 1: Wi-Fi hotspot scenario.

The 802.11 standard defines a Power-Saving Mode (PSM), aimed at reducing the energy consumption of mobile devices. Recently, several works have been devoted to highlight PSM limitations and propose enhancements. The most closely related to our work are STPM [1], BSD [30] and SPSM [38] (we provide a comprehensive survey of the related work in Section 2). These works highlight that PSM adds high transfer delays in a range of application and network configurations. Besides decreasing the user QoS, this might even *increase* the energy consumption of the device as a whole, with respect to the case when PSM is not used: the energy saved on the wireless interface by PSM gets overwhelmed by the energy spent by the rest of the mobile device during the additional transfer time. In this paper we focus on a different – yet very popular – scenario (thoroughly described in Section 3) where the PSM additional delay is fairly limited. In Section 4 we show that using PSM in this scenario is an advisable choice. Thus, in Sections 5 and 6 we extensively characterize the PSM performance for a wide range of key parameters. To the best of our knowledge, this is the first work providing such a detailed PSM analysis. We show that PSM is very effective during traffic *burst*. With respect to the case when it is not used, PSM is able to save up to 90% of the energy required to download a burst. However, PSM is not quite fit to deal with long *User Think Times* between bursts, that can actually represent the main source of energy consumption. From this standpoint, the original contribution of our work consists in a deep exploration on how to further reduce the energy consumption during User Think Times. Specifically, in Section 7 we define a *Cross-Layer Energy Manager* (XEM) that exploits information scattered across several layers in the protocol stack to detect the beginning of User Think Times and bursts. During bursts XEM activates PSM, while during User Think Times it switches the wireless interface off. XEM does not degrade the user QoS, and achieves additional energy saving with respect to the standard PSM between 20% and 90%, depending on the User Think Time length, and the bursts' size.

In this work we provide two contributions. In the first part of the paper, we provide an accurate model of the 802.11 PSM, and deeply characterize the 802.11 PSM performance. In comparison with existing works, which usually highlight scenarios in which PSM is not effective, we show that there is a broad range of cases in which PSM can be successfully used to reduce the energy consumption. In the second part of the paper, we turn to analyze PSM inefficiencies, and propose and evaluate XEM. With respect to existing work, XEM smoothly integrates with current 802.11 PSM and does not require any modification of legacy protocols and applications. XEM is thus a very lightweight, yet efficient, solution to improve PSM in cases in which it is not efficient.

2 Related work

Understanding and enhancing the performance of wireless LANs, mainly in terms of energy saving, has deserved increased attention in the last few years. Papers in this field can be divided into two main categories. Some works highlight limitations of PSM and propose possible enhancements. Other works propose energy-management policies that are not specifically tailored to 802.11 but can be applied to this technology, as well. For ease of reading, in the following of this section we follow the above classification. For the sake of space, and because the environment is

significantly different, we do not survey the broad research area on energy management for ad hoc networks.

2.1 Energy-management policies for Wi-Fi hotspots

A pioneering work on this topic is presented by Krashinsky and Balakrishnan in [30]. They carry out a simulation analysis of PSM in presence of Web-browsing traffic. In particular, they consider a single mobile user (i.e., no contention) inside the hotspot. The authors of [30] show that PSM can save around 90% of the energy spent by the wireless interface at the cost of highly increased delay in the Web-page downloads. To cope with this problem, they propose the Bounded Slowdown Protocol (BSD). In BSD, the mobile host listens the Access Point Beacons with decreasing frequency during idle times, to be mostly sleeping during User Think Times. BSD trades off energy consumption for lower additional delays. Specifically, if the maximum acceptable delay is very low, BSD actually consumes more energy than PSM. Therefore, BSD can be more or less suitable than PSM, if the additional delay or the energy consumption deserves more importance for the user. As noted in the paper, BSD focuses on a scenario where PSM tremendously increases the transfer delay, and thus it represents a very effective solution. However, in our scenario this additional delay is quite limited. It should be noted that, thanks to its flexibility, our Cross-Layer Power Manager (XEM) is able to use either PSM or BSD during bursts. In contrast to BSD, XEM switches the wireless interface off during User Think Times. As discussed in Section 6.3, the XEM ability to distinguish interarrival times (for which it is more convenient using the sleep state) from User Think Times (for which it is better to switch the wireless interface off) grants greater energy saving.

More recently, Qiao and Shin proposed the Smart Power-Saving Mode (SPMS) [38]. SPMS can be seen as a BSD enhancement. During an idle time, BSD defines *statically* the set of points in time where the mobile host listens for Access Point Beacons. Instead, SPMS defines this set of points dynamically, based on an estimate of the idle time duration. SPMS is more energy efficient than BSD, and still achieves the same performance in bounding the additional delay. However, it still consumes more energy than PSM in some cases, and just exploits the sleep state of the wireless interface to conserve energy. Since SPMS is close to BSD in spirit, the same remarks discussed above apply to SPMS, as well.

The authors of [34] propose the *Dynamic Beacon Period* algorithm (DBP). As BSD and SPSM, DBP aims at reducing the additional delay introduced by PSM to Web-page download times. Basically, each mobile host selects its own Beacon Interval, and the Access Point is responsible for generating (custom) Beacon frames for each mobile host. Several scalability issues, that are key points to fairly evaluate DBP, are not addressed in [34]. As in the cases of BSD and SPMS, DBP just exploits the sleep state of the wireless interface to conserve energy, for any kind of idle time that might occur.

Anand et al., [1] carry out an experimental evaluation of PSM both on PDAs and laptops. They primarily focus on the traffic generated by applications using network file systems such as NFS and Coda. Their results confirm the conclusions in [30], as far as the additional delay introduced by the PSM. To overcome this problem, they propose the Self-Tuning Power Management (STPM) protocol. STPM operates at the Operating System level, and exploits *hints* provided by the network applications. Essentially, hints describe the near future requirements of applications in terms of networking activities. STPM exploits these hints, and the energy characteristics of the entire system, to manage the wireless interface appropriately. When these hints are not available, STPM estimates the traffic patterns by spoofing it. Like STPM, our Cross-Layer Energy Manager sits on top of different energy management policies, and dynamically chooses the most appropriate one. The main difference between [1] and our work is that XEM is simpler, and never requires collaboration from the applications, i.e., no modifications of the application code is required. Again, [1] focuses on a scenario where PSM delays are a big problem, while in our scenario they are not.

Finally, [9, 39] propose energy-management policies for 802.11 WLAN that are orthogonal to the work presented in this paper, and hence can coexist with XEM.

2.2 Energy-management policies for generic wireless LANs

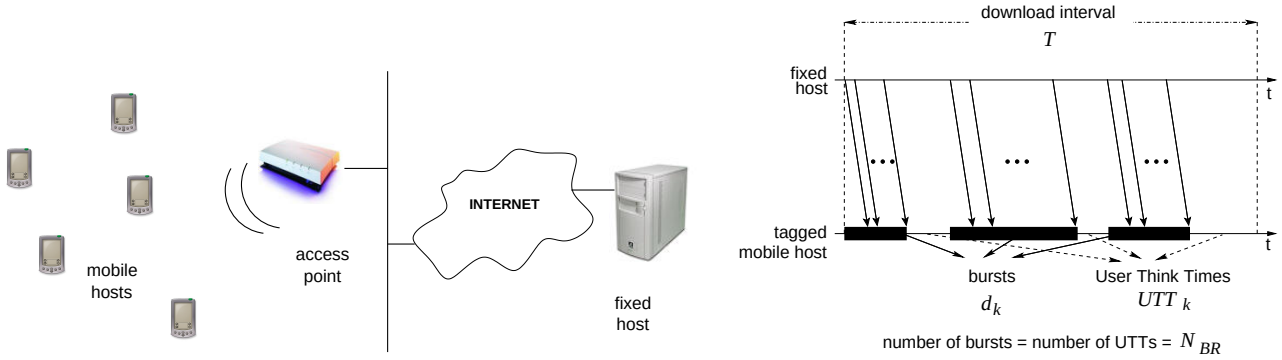
Other works face the energy-management problem in WLAN environments, but do not focus on a specific wireless technology. The authors of [32] propose a solution entirely centralized at the Access Point. Time is divided in Beacon Intervals (as in the standard 802.11), and – at the beginning of each Beacon Interval – the Access Point computes a schedule for transmitting frames during the coming Beacon Interval. Before any other transmission, the Access Point broadcasts a Beacon Frame to publicize which mobile hosts are going to receive frames. These mobile hosts remain awake until they have received all the scheduled frames, while the other hosts can immediately switch to a low-power mode. This solution gives to the Access Point the flexibility of implementing several scheduling policies, but requires i) significant computational burden at the Access Point, and ii) non-trivial modifications to the 802.11 standard.

The authors of [6] design an energy manager tailored exclusively to Web-based applications. By means of prefetch-like techniques, Web pages are transferred over the WLAN in a single (or few) burst, thus maximizing the amount of time during which the wireless interface is switched off. This technique does not introduce significant additional delays. Of course, the energy manager is tied with the particular application it is designed for. In [3] it is shown that this constraint can be relaxed with an acceptable degradation of the energetic performance. Specifically, [3] dynamically estimates the expected duration of idle times. The mobile host is switched off for the (predicted) duration of the idle time. The work in [6] and [3] inspired some ideas on how User Think Times and new bursts can be detected. However, XEM fully exploits PSM when appropriate, and can avoid relying on application-level information.

The works in [31, 44, 46] use inactivity timeouts to decide when to switch off the wireless interface. Timeout values are fixed, and depend on the specific application. [31] relies on an Indirect-TCP architecture and buffers at the Access Point packets arriving while the mobile host is disconnected. Instead, [44] avoids any support from the Access Point, and exploits knowledge of the application behavior to avoid missing packets. Also [46] uses a pure client-centric approach, i.e., no support from the Access Point is exploited. Specifically, [46] uses an approach very similar to [3], in the sense that interarrival times are estimated on-line. Furthermore, inactivity timeouts are used to detect User Think Times. With respect to [3], no support from the Access Point is exploited. Hence, packets that may arrive while the mobile host is disconnected are lost. Inactivity timeouts are also used by XEM. However, in our system they are dynamically adjusted based on the status of the network path.

The works in [33, 17, 42] advocate energy management at the operating system level. [33] exploits on-line application-level hints to decide when to shut down the wireless network. Hence, this system requires modifications to the application code. The authors of [17, 42] formulate the energy-management problem as a linear program, where the objective is minimizing the energy consumption of a particular component, and the maximum tolerable performance degradation (for example in terms of additional delay) is the constraint. Then, they derive optimal energy management policies to drive the component in the different operating modes. The main drawback of this approach is that it requires a-priori statistical models of the component usage. This information is not required by XEM.

Finally, other approaches to energy management include transmission power control techniques [22], or a drastic re-design of the application-level architecture [37, 24, 25]. Specifically, [25] introduces a quite recent technology, named AJAX. The main idea is decoupling (in Web-like applications) the user and the server via a proxy-based component (called Ajax engine) running on the client. The user actually interacts with the Ajax engine, that asynchronously fetch from the server the data required to fulfill the user requests. Ajax is able to significantly reduce the amount of data exchanged over the network in case of slight modification of a currently-rendered Web page. In general, application-level techniques are orthogonal to XEM. In the case of AJAX, the most straightforward interaction we can see is using XEM below AJAX. XEM would interpret the traffic pattern generated by AJAX (instead of that generated by the user), and manage the wireless interface accordingly.



(a) Wi-Fi hotspot scenario (b) Application-level traffic

Figure 2: The reference environment.

3 Detailed Scenario and Evaluation Methodology

3.1 Reference Scenario

In our analysis, we consider the typical Wi-Fi hotspot scenario, depicted in Figure 1 and replicated in Figure 2(a) for the reader convenience, in which a mobile user accesses the Internet through an Access Point. We focus on best-effort Internet applications, such as Web browsing, e-mail, file transfer (hereafter referred to as *reference applications*). This choice is motivated by the evidence that the traffic generated by these applications represents the lion's share of the today Internet traffic, and they are very likely to be the dominant applications also in the near-future Internet [10].

Figure 2(b) shows a snapshot of the typical traffic generated by the reference applications. A *tagged mobile host* downloads a predefined number of *bursts* (N_{BR}) from a fixed server connected to the Internet. The download of two consecutive bursts is separated by a *User Think Time* (UTT) during which no traffic flows between the server and the mobile host (the other details shown in the Figure are related to the PSM model, and will be thus explained in Section 5). Though very simple, this traffic model captures the typical user behavior for non real-time applications. For example, Web users download a page (i.e., a burst) and then read the page contents without generating any traffic on the network. By considering several such downloads (say, N_{BR}) from the same site, we model the behavior of a user navigating a single Web site for a while. Because of its importance, some concepts of the paper are presented by using the Web as the reference application. However, the traffic model is general enough to represent also other best-effort applications, as well. Furthermore, we investigated a broad range of the scenario parameters when evaluating PSM and XEM. Therefore, we believe our results are not valid only in the Web case. Furthermore, as far as the XEM definition, it is not heavily tied to the Web case, and can work with different applications, as discussed in Section 7.4.

We assume that the mobile host communicates with the fixed server through a standard TCP-Reno (without delayed acks [45]) connection. We also assume that consecutive bursts be downloaded over the same connection. In the Web case this corresponds to using the persistent-connection option defined by HTTP/1.1 [27]. Since HTTP file transfers usually consist of few KB [13, 20, 21] this option was defined to avoid the huge overhead of opening a new TCP connection for each file transfer. It reduces download times, and allows TCP to precisely learn the path congestion. In Section 4 we highlight that this option has further advantages when PSM is used. Specifically, the additional delay introduced by PSM to TCP transfers becomes fairly small, and it does not significantly impact on the energy consumption of the device as a whole. Thus, using persistent connections is a good idea for the other reference applications, as well. We finally assume that the mobile host does not utilize parallel concurrent connections to download bursts. This is aligned with the suggestions of [27] when persistent connections are used,

and it makes the analysis of both PSM and XEM simpler. In Section 7.4 we highlight how XEM can be extended to work in the case of concurrent TCP connections. One might argue that anyway the legacy TCP/IP architecture exhibits poor performance in a WLAN environment, both in terms of throughput and energy consumption [12]. However, TCP/IP is currently the only off-the-shelf solution for Wi-Fi hotspots and thus our environment is similar to real-world WLAN installations.

In our scenario, the hotspot is populated by other N (background) mobile hosts in addition to the tagged mobile host. We assume that, at each point in time, M mobile hosts out of N are active, i.e., they have a frame ready to be sent. As discussed in Section 6.2.3, by varying the number of active mobile hosts (i.e., M) we can analyze the sensitiveness of PSM to the contention level in the hotspot, and – therefore – its scalability with respect to the number of users sharing the same Access Point.

3.2 802.11 Power-Saving Mode (PSM)

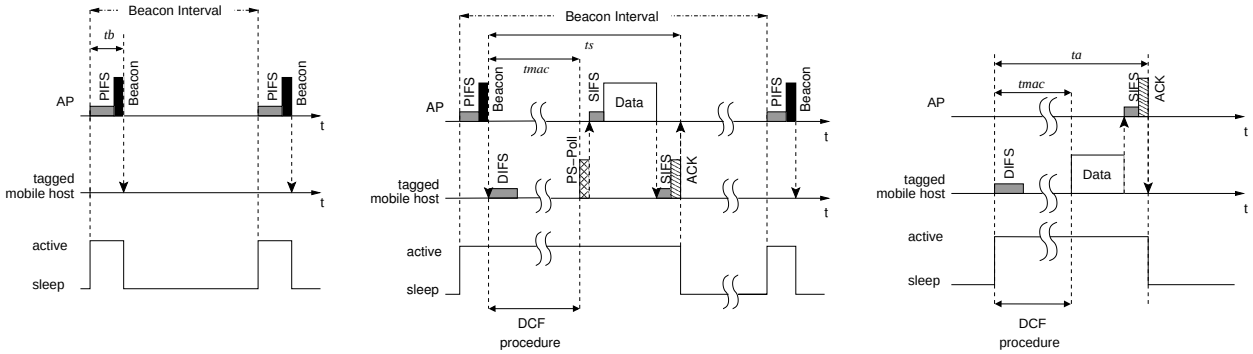
As a significant part of this work is devoted to analyze the PSM performance, in this section we briefly recall the main features of this algorithm. The interested reader is referred to the IEEE 802.11 standard for a complete description [29]. The objective of the 802.11 PSM is to let the wireless interface of a mobile host in the active mode only for the time necessary to exchange data, and turn it in sleep mode whenever it becomes idle. In a Wi-Fi hotspot, this is achieved by exploiting the central role of the Access Point. Each mobile host within the hotspot lets the Access Point know whether it utilizes the PSM or not. Since the Access Point relays every frame from/to any mobile host, it buffers the frames addressed to mobile hosts using the Power-Saving Mode. Every Beacon Interval – usually, 100 ms –, the Access Point broadcasts a special frame, named Beacon (Figure 3(a)). This frame contains a Traffic Indication Map (TIM) that indicates PSM mobile hosts having at least one frame buffered at the Access Point. PSM mobile hosts are synchronized with the Access Point, and wake up to receive Beacons. If they are indicated in the TIM, they download the frames as is shown in Figure 3(b). Specifically, the PSM mobile host sends a special frame (ps-poll) to the Access Point by means of the standard DCF procedure. Upon receiving a ps-poll, the Access Point sends the first data frame to the PSM mobile host, and receives the corresponding ack frame. If appropriate, the Access Point sets the More Data bit in the data frame, to announce other frames to the same PSM mobile host. To download the next frame, the mobile host sends *another* ps-poll. When, eventually, the mobile host has downloaded all the buffered frames, it switches to the sleep mode.

To send a data frame, a PSM mobile host (if the case) wakes up and performs the standard DCF procedure. Specifically, the PSM mobile host sends the data frame, and receives an ack frame from the Access Point (Figure 3(c)).

To summarize, a mobile device operating in Power-Saving Mode is required to be awake to perform *three* basic operations: (i) receiving Beacon frames; (ii) downloading data frames from the Access Point; and (iii) sending data frames to the Access Point. This remark is fundamental for the analytical characterization of PSM that have been derived in [4] presented in Section 5.

3.3 Evaluation Methodology

In the environment described above, one of the main inefficiencies in energy usage is *listening during idle times*. It is well known that the traffic generated by the reference applications exhibits different types of idle times [21, 20, 13, 8]. Specifically, idle times *inside* traffic bursts (referred to as *interarrival times*) are typically very short, less than 1 s [21, 20, 13]. On the other hand, idle times *between* consecutive bursts (referred to as *User Think Times*), are longer and may last up to 60 s and beyond [21]. As the goal of PSM is reducing the energy consumption during idle times, we extensively analyze its behavior with respect to both interarrival times and User Think Times.



(a) receiving beacons (b) receiving frames (c) sending frames

Figure 3: PSM operations.

The amount and duration of interarrival times is dictated by both the application-level protocols, and the characteristics of the network path between the mobile and the fixed host. Thus, we analyze the PSM performance with respect to key applications and network parameters, i.e., i) the average burst size; ii) the transport-level throughput; and iii) the MAC-level contention (i.e., the number of users in the same Wi-Fi hotspot).

We then study the PSM performance during User Think Times. We show that the energy consumption during these phases may dominate the energy consumption due to the whole traffic pattern. Since PSM is far from optimal during UTTs, we define and evaluate a Cross-Layer Energy Manager (XEM) that drastically reduces PSM energy consumption during UTTs.

Our analysis relies on both analytical and simulation results. Specifically, we extend the simulation model used in [16] to implement the reference network scenario described above (see Section 6 for details). Furthermore, to better understand the PSM behavior shown by simulation, we exploit an analytical model[2] we derived in [2, 4, 36], that provides closed formulas for the energy consumption in the cases where PSM is used or not. A brief presentation of this model is given in Section 5.

3.3.1 Performance Indices

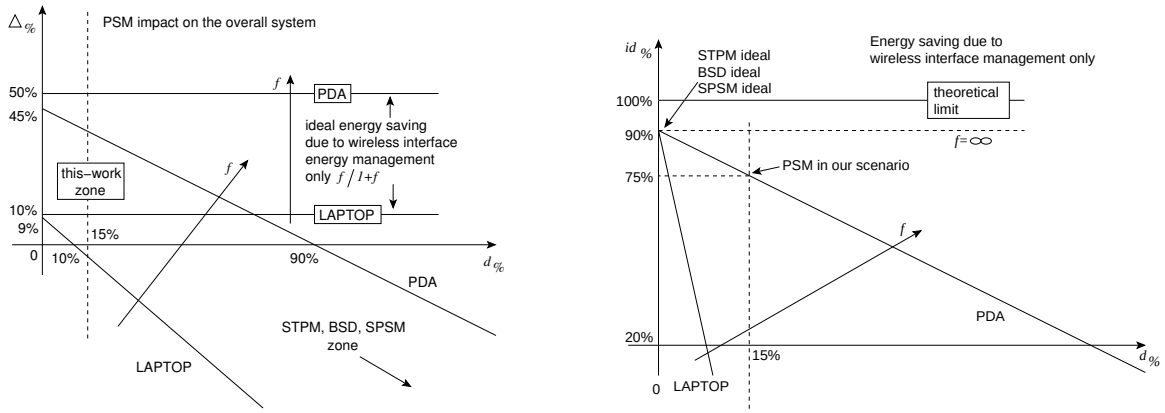
Our analysis is mainly based on the following performance figures:

- E_C: the average amount of energy spent by the wireless interface to download N_{BR} bursts from the fixed to the tagged mobile host, when PSM is not active (i.e., in continuous active mode, CAM);
- E_P: the average amount of energy spent by the wireless interface to download N_{BR} bursts from the fixed to the tagged mobile host, when PSM is active;
- $R(E_P, E_C)$: the ratio between the above indexes. This is a key index, since it shows the fraction of energy spent when PSM is active, with respect to the case when no energy management is used, and, thus, it shows the PSM efficiency.

Throughout the paper, we analyze energy consumption breakdowns for the different energy-saving policies under investigation. In that cases, more specific performance indexes are defined. When meaningful, the index $R(\cdot, \cdot)$ is also applied to couples of those indexes, to show the relative advantage of the first one with respect to the second one.

Admittedly, most of our analysis neglects the energy consumption of mobile-device components other than the wireless interface. Results discussed in the next section show that in our scenario this is a reasonable choice.

[2] As shown in [2, 4, 36], analytical and simulation results fully agree.



(a) (b)

Figure 4: PSM impact on the overall system (a), and on the wireless interface only (b).

4 Effects of PSM delay on energy consumption

Thanks to the results in [1, 30, 38] it is now well understood that the additional delay introduced by PSM to TCP transfer times may even make PSM counterproductive from an energy-saving standpoint. Even though much energy spent on the wireless interface can be saved by PSM, the other device components continue to drain energy during the additional transfer time. This cost may completely overwhelm the energy saved on the wireless interface. For example, Anand et al. [1] measure slowdown factors as high as 16x to 32x when using PSM.

It can be noted that high additional delays arise when *short* TCP connections are used over *short* RTT paths. Indeed, [1, 30, 38] measure high additional delays in cases where the RTT between the mobile and fixed hosts (measured when PSM is not active) is very short (few tens of milliseconds). Moreover, they focus on Web traffic *without* persistent connections [30], and NFS-like traffic [1][3], which actually generate very short TCP connections. Due to the TCP 3-way handshake, and to the slow-start algorithm, a new TCP connection requires several RTTs even to fetch a few KBytes. Furthermore, the work in [30] shows that PSM rounds every RTT up to the next 100ms, due to the beaconing mechanism. This can be a very high additional delay for short RTTs (i.e., in the order of few tens of milliseconds). Therefore, when PSM is used to download data over short-lived, non-persistent TCP connections, in case of short RTTs, the additional delay can be very high.

We purposely choose a different scenario for our analysis. As mentioned in Section 3, we consider persistent TCP connections, as suggested by HTTP/1.1 [27]. Furthermore, we focus on a broader RTT range (measured when PSM is not active), in the order of few hundreds of milliseconds. Even though Web proxies and caches tend to reduce the RTT, they cannot be used with any Web content (e.g., cannot be used with dynamically generated pages). Moreover, it is still common to measure RTTs in the order of 200 ms and above while accessing popular servers over intercontinental paths (e.g., accessing ebay.com, cnn.com, nasdaq.com, amazon.com between Europe and US). In this case, fetching data requires less RTTs, because the congestion window is already stable, since the connection is persistent. Furthermore, the cost of rounding up every RTT to the next 100ms is reduced if the original RTT (measured without PSM) is already a few hundreds of milliseconds. Indeed, in our scenario the additional delay that we have measured, averaged over all the experiments presented in the following, is just around 15% of the original transfer time (corresponding to a 1.15x slowdown).

To understand the impact of this slowdown on energy consumption, we follow a simple analytical approach. When PSM is not used, we assume that both the wireless interface and the rest of the system constantly drain a

[3] Anand et al. in [1] focus on other types of traffic as well. However, all the traffic patterns for which PSM introduces high delays share the same features.

fixed amount of power, denoted as $P^{(N)}$ and $P^{(B)}$, respectively. Let us denote by t the download time of a burst, and by $e_C^{(N)}$ and $e_C^{(B)}$ the energy spent in continuous active mode during t by the wireless interface and the rest of the system, respectively. Let us finally denote by f the relative cost of the wireless interface with respect to the rest of the system, i.e. $f = P^{(N)}/P^{(B)}$. Thus, the energy spent in the burst download is $e_C = e_C^{(N)} + e_C^{(B)} = P^{(N)}t + P^{(B)}t = (1+f)e_C^{(B)}$. The use of PSM has two effects. On the one hand, it reduces $e_C^{(N)}$. Let us denote this energy saving by β, i.e., $e_P^{(N)} = \beta e_C^{(N)}$ where $e_P^{(N)}$ is the energy spent on the wireless interface when PSM is used. On the other hand, PSM increases the energy of the rest of the mobile host because of the additional delay. If d denotes this additional delay as a fraction of t, then we obtain $e_P^{(B)} = (1+d)tP^{(B)} = (1+d)e_C^{(B)}$. It is now easy to evaluate the overall energetic advantage brought by PSM. Specifically, we define the index Δ as $\Delta = (e_C - e_P)/e_C$, where $e_P = e_P^{(N)} + e_P^{(B)}$. After simple manipulations we obtain $\Delta = 1 - \frac{f\beta+1+d}{1+f}$.

Figure 4(a) plots Δ as a function of d for various values of f. Typically, f increases as the device form-factor shrinks; representative values for a laptop and a PDA are 1/9 and 1, respectively [1]. Characterizing β is the task of most part of this paper. However, we can here anticipate that $\beta = 0.1$ is a reasonable value to have a first rough – yet significant – picture. This value also matches other results in the literature [30, 38]. Figure 4(a) clearly differentiates our work from [1, 30, 38]. STPM [1], BSD [30] and SPMS [38] are mainly designed to operate in cases when the additional delay is large (e.g., the 16x slowdown measured by [1] corresponds to $d = 1500\%$!). Indeed, in these cases Δ drops below 0, stating that PSM actually produces an energy increase on the whole device. Instead, our work focuses on a region where d is limited, and PSM becomes effective, mostly for small form-factor devices. For this class of devices, PSM saves a large portion of the energy consumption due to networking activities, without charging significantly the other device components. Based on these results, hereafter we measure the energy consumption of the wireless interface.

Figure 4(a) shows the theoretical limits achieved by an ideal policy that completely eliminates the wireless interface energy consumption (this policy clearly represents the asymptotical limit of any energy management technique focused on the wireless interface). On a burst download lasting t seconds, the ideal policy consumes $e_I = e_C^{(B)}$. Figure 4(b) compares more thoroughly the PSM performance with e_I. Specifically, it plots the index id, which is defined as the ratio between the energy saved by PSM, and the energy saved by the ideal policy, i.e., $id = (e_C - e_P)/(e_C - e_I) = 1 - \beta - \frac{d}{f}$. In our scenario, PSM achieves 75% of the ideal energy saving. It is interesting to note that the additional delay reduces the energy saving just by 15%. Furthermore, the id index can be used also to roughly understand the maximum expected improvement of STPM, BSD and SPMS over PSM in our scenario. STPM activates or deactivates PSM based on predictions about the future traffic profile. This way, it reduces the additional delay to negligible values. In the best possible case, it spends on the wireless interface the same energy spent by PSM, without increasing the energy consumption of the rest of the device. Setting $d = 0$ in the id formula thus gives the maximum energy saving of STPM. Getting analytical results for BSD and SPSM is not straightforward. However, by inspecting the results provided in [30, 38] we can still derive some limit. SPSM is generally able to avoid the additional delay, and in several cases achieves the same wireless interface energy consumption of PSM. Thus, the id value for $d = 0$ is a good indication for the maximum SPSM performance, as well. Also BSD is able to reduce the additional delay to negligible values. [30] shows that this is often achieved without increasing the wireless interface energy consumption with respect to PSM. We thus consider the same maximum energy saving for BSD, as well. These optimal values are indicated in Figure 4. STPM, BSD, and SPSM seem able to improve the performance of PSM also in our scenario. Nevertheless, we believe that PSM still represents a valid option, because i) it achieves significant energy saving anyway, ii) it is already available on most of the commercial devices, and iii) it is thus a free-of-charge solution. It should also be noted that the *real* performance of STPM, BSD and SPSM can be lower than the values in Figure 4(b). As an example, [30, 38] show that, in order to eliminate the additional delay, BSD might increase the energy spent on the wireless interface with respect to PSM. On the other hand, in order to keep the same energy saving, BSD must introduce additional delays around 14%. Similar remarks suggest that,

though being very effective when d is high, BSD, STPM and SPSM do not perform far from PSM in our scenario. As a final remark, it should be noted that the above discussion applies to the burst download phases. We postpone a similar discussion about UTTs to Section 7, to have the chance of including also XEM in the picture.

The above remarks show that there is a broad range of cases in which using PSM as an energy-saving technique is advisable. Therefore, we now analyze in depth the PSM performance in terms of energy saving.

5 Analytical model

With reference to the network scenario and the application-level traffic model depicted in Figure 2(a) and Figure 2(b), respectively, in this section we derive a model for evaluating the average energy spent by the tagged mobile host to download N_{BR} bursts from the fixed host (any two consecutive bursts are separated by a User Think Time). Due to space reasons, we here present the main analytical results, and we skip many detailed proofs. Interested readers can refer to [2, 4, 36] for all the details.

To model the tagged mobile-host behavior we utilize the following approach. We replicate n times the download of N_{BR} bursts, and focus on the generic i-th replica. $E_P^{(i)}$ and $E_C^{(i)}$ denote the energy spent during the i-th replica when PSM is enabled and disabled, respectively. In the following, we derive closed formulas for $E_P^{(i)}$ and $E_C^{(i)}$, and show that $\left\{E_P^{(i)}\right\}_{i,\ldots,n}$ and $\left\{E_C^{(i)}\right\}_{i,\ldots,n}$ are composed by i.d. random variables[4]. Therefore, we can express E_P and E_C as follows:

$$\begin{cases} E_P &= \lim\limits_{n\to\infty} \dfrac{\sum_{i=1}^{n} E_P^{(i)}}{n} \\ E_C &= \lim\limits_{n\to\infty} \dfrac{\sum_{i=1}^{n} E_C^{(i)}}{n} \end{cases} \tag{1}$$

By introducing the closed formulas for $E_P^{(i)}$ and $E_C^{(i)}$ in Expressions 1, we finally obtain the closed formulas for E_P and E_C.

As far as $E_C^{(i)}$, it is worth noting that, when PSM is disabled, the wireless interface of the tagged mobile host is always active. Hence, if $T^{(i)}$ denotes the duration of the i-th replica (also referred to as the *download interval*), and P_{ac} denotes the power drained by the tagged mobile host in the active mode, $E_C^{(i)}$ can be expressed as

$$E_C^{(i)} = T^{(i)} \cdot P_{ac} \ . \tag{2}$$

On the other hand, when PSM is enabled, the tagged mobile host remains active just for a portion ($T_{ac}^{(i)}$) of the download interval[5], while it is sleeping for the rest of the time ($T_{sl}^{(i)}$). Therefore, if P_{sl} is the power drained by the tagged mobile host in the sleep mode, $E_P^{(i)}$ can be expressed as follows:

$$E_P^{(i)} = T_{ac}^{(i)} \cdot P_{ac} + T_{sl}^{(i)} \cdot P_{sl} = T_{ac}^{(i)} \cdot (P_{ac} - P_{sl}) + T^{(i)} \cdot P_{sl} \ . \tag{3}$$

Equations 2 and 3 show that both $E_P^{(i)}$ and $E_C^{(i)}$ depend on $T^{(i)}$ and $T_{ac}^{(i)}$. In the following subsections we derive $T^{(i)}$ and $T_{ac}^{(i)}$, respectively.

5.1 Modeling the download interval (T)

With reference to a generic i-th replica, $T^{(i)}$ may be thought of as made up of two components (see Figure 2(b)): i) the total time during which bursts are downloaded ($T_{data}^{(i)}$), and ii) the total inactive time due to User Think Times ($T_{idle}^{(i)}$). Denoting by $td_k^{(i)}$ the time required by the tagged mobile host to download the k-th burst in the i-th replica, and by $UTT_k^{(i)}$ the duration of the k-th User Think Time in the i-th replica, $T^{(i)}$ can be written as follows:

[4] To simplify the notation, in the following we omit indicating the range of variability of i, e.g. $\left\{E_P^{(i)}\right\}_{i,\ldots,n}$ is referred to as $\left\{E_P^{(i)}\right\}$

[5] $T_{ac}^{(i)}$ also includes the transition times from the sleep to the active mode.

$$T^{(i)} = T^{(i)}_{data} + T^{(i)}_{idle} = \sum_{k=1}^{N^{(i)}_{BR}} td^{(i)}_k + \sum_{k=1}^{N^{(i)}_{BR}} UTT^{(i)}_k \ . \tag{4}$$

It can be shown that $\left\{T^{(i)}_{data}\right\}$ and $\left\{T^{(i)}_{idle}\right\}$ are composed by identically distributed random variables. Furthermore, for each couple $\langle i,k\rangle$, $N^{(i)}_{BR}$ and $td^{(i)}_k$, as well as $N^{(i)}_{BR}$ and $UTT^{(i)}_k$, are mutually independent. It is also worth pointing out that we assume that TCP always works in the steady state (i.e., we do not consider slow-start phases), which is a common assumption in the literature [35], and very reasonable in the case of persistent connections. Furthermore, to simplify the analysis, we approximate the steady-state TCP throughput with a constant value[6], hereafter referred to as γ_{TCP}. Therefore, if $E[d]$ denotes the average burst size, after simple manipulation the average value of the download interval can be expressed as:

$$E[T] = E[N_{BR}] \cdot \left\{\frac{E[d]}{\gamma_{TCP}} + E[UTT]\right\} \ . \tag{5}$$

5.2 Modeling the time spent in the active mode (T_{ac})

Since we are assuming a TCP/IP architecture, the traffic on the WLAN related to the tagged mobile host includes: i) TCP segments[7] coming from the fixed server; ii) TCP acks sent by the tagged mobile host to the fixed server; and iii) Beacon frames periodically broadcast by the Access Point. Thus, $T^{(i)}_{ac}$ is the time spent in the active mode by the tagged mobile host to handle these traffic components. Before proceeding on, we need to introduce some assumptions and emphasize some properties related to our model.

Property 1. In the following, we assume that Beacon frames are safely transmitted, i.e., they do not collide with transmissions from any other mobile host. In other words, the MAC protocol guarantees that the shared medium is idle at the beginning of each Beacon Interval, and no transmissions are attempted until the Beacon frame is received (see Figure 3(a)). This assumption is aligned with the most up-to-date proposals within the 802.11 working groups [28].

Property 2. Let us define a *sequence of frames* as a set of frames exchanged between the mobile host and the Access Point, where each frame is spaced from the previous one by a SIFS interval. According to the 802.11 DCF definition [29], in our WLAN environment only the first frame of a sequence can undergo collision. In other words, either the first frame of a sequence collides, or the whole sequence is safe.

Property 3. Each TCP segment sent by the fixed host to the tagged mobile host is encapsulated into a distinct IP packet. Thus, if we assume that both IP- and 802.11 MAC-level fragmentation are disabled, the tagged mobile host downloads each TCP segment from the Access Point inside a distinct data frame. Since we also assume that the RTS/CTS mechanism is disabled, downloads occur by exchanging a sequence of frames including a ps-poll, data, and ack frame (between the tagged mobile host and the Access Point), as shown in Figure 3(b). Similarly, each TCP ack is uploaded to the Access Point inside a distinct data frame, i.e., by exchanging a sequence of frames composed by a data and an ack frame (see Figure 3(c)).

Property 4. Let i) $s^{(i)}_j$ be the time required by the tagged mobile host to download the generic j-th TCP segment during the i-th replica, starting from the point in time when the tagged mobile host starts the DCF procedure to send the related ps-poll frame; ii) $a^{(i)}_r$ be the interval required by the tagged mobile host to upload the r-th TCP ack during the i-th replica, starting from the point in time where the tagged mobile host starts the DCF procedure to send the related data frame; and iii) $b^{(i)}_l$ be the time required by the tagged mobile host to receive the l-th Beacon frame during the i-th replica, starting from the beginning of the related Beacon Interval. Then, for any triple $\langle j,r,l\rangle$,

[6] The validation of the analytical model carried out in [2, 4, 36] shows that these assumptions do not compromise the accuracy of the analytical results.

[7] For the sake of simplicity, we indicate TCP segments containing application data as TCP segments, while TCP acks denote TCP segments containing just acknowledgments.

it can be shown that the time intervals $s_j^{(i)}$, $a_r^{(i)}$ and $b_l^{(i)}$ do not overlap.

Based on the above properties, $T_{ac}^{(i)}$ can be regarded as the sum of times required to i) receive all the Beacon frames, ii) download all the TCP segments, and iii) upload all the TCP acks, within a download interval i.e.

$$T_{ac}^{(i)} = \sum_{j=1}^{N_{seg}^{(i)}} s_j^{(i)} + \sum_{r=1}^{N_{ack}^{(i)}} a_r^{(i)} + \sum_{l=1}^{N_b^{(i)}} b_l^{(i)} \ . \tag{6}$$

In (6), $N_{seg}^{(i)}$, $N_{ack}^{(i)}$ and $N_b^{(i)}$ are the number of TCP segments, TCP acks, and Beacon frames exchanged (between the mobile host and the Access Point) during the i-th replica, respectively. In our model: i) the number of TCP segments downloaded is equal to the number of TCP acks uploaded (i.e., $N_{seg}^{(i)} = N_{ack}^{(i)}$); and ii) $b_l^{(i)}$ can be reasonably approximated with a constant value, throughout referred to as b. Finally, by analyzing the properties of the random variables $N_{seg}^{(i)}$, $N_{ack}^{(i)}$, $s_j^{(i)}$ and $a_r^{(i)}$, it can be shown that the average value of T_{ac} can be expressed in a very intuitive way:

$$E\left[T_{ac}\right] = E\left[N_{seg}\right] \cdot \left(E\left[s\right] + E\left[a\right]\right) + E\left[N_b\right] \cdot b \ . \tag{7}$$

Deriving closed formulas for $E\left[s\right]$ and $E\left[a\right]$ would require a detailed analysis of the 802.11 DCF function. The complete model accounting for *all* the DCF details (retransmissions, contentions, etc.) is derived in [2, 4, 36], and is here omitted for the sake of space. The main issue to be highlighted here is that both $E\left[s\right]$ and $E\left[a\right]$ include two components, i.e. the *average MAC delay* and the *average sequence time* (see Figure 3(b,c)). The average MAC delay is defined as the interval between the time when the DCF procedure is invoked to transmit a frame, and the time of the successful transmission (possibly after a number of unsuccessful attempts). Intuitively, this component is statistically equivalent for both TCP segments and TCP acks, and corresponds to the time spent by the tagged mobile host in the DCF procedure to send the ps-poll frame and the data frame containing the TCP ack, respectively (see Figure 3(b,c)). The average sequence time is defined as the interval between the time when the transmission of the first frame in a sequence starts, and the time when the reception of the last frame in that sequence ends. Clearly, the average sequence time depends on the frames in the sequence, and is thus different for TCP segments and TCP acks (see Figure 3(b,c)).

The last step to derive a closed formula for $E\left[T_{ac}\right]$ is evaluating $E\left[N_{seg}\right]$ and $E\left[N_b\right]$. It can be shown that the average number of TCP segments exchanged during a download interval ($E\left[N_{seg}\right]$) is equal to the average size of all bursts downloaded in that replica, divided by the Maximum Segment Size (MSS) of the TCP connection, i.e.,

$$E\left[N_{seg}\right] = \frac{E\left[N_{BR}\right] \cdot E\left[d\right]}{MSS} \ . \tag{8}$$

In addition, the average number of Beacon frames received by the tagged mobile host during a replica ($E\left[N_b\right]$) is the ratio between the average download-interval duration, $E\left[T\right]$, and the duration of a Beacon Interval, BI:

$$E\left[N_b\right] = \frac{E\left[T\right]}{BI} \ . \tag{9}$$

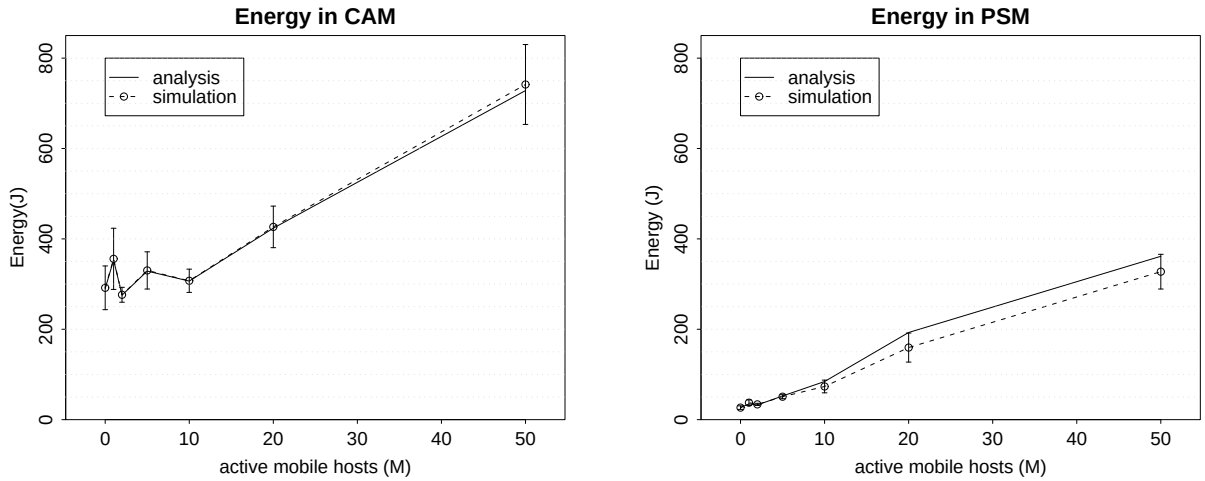
By substituting Equations 8 and 9 into Equation 7 we obtain the following closed formula for $E\left[T_{ac}\right]$:

$$E\left[T_{ac}\right] = \frac{E\left[N_{BR}\right] \cdot E\left[d\right]}{MSS} \cdot \left(E\left[s\right] + E\left[a\right]\right) + \frac{E\left[T\right]}{BI} \cdot b \ . \tag{10}$$

Finally, by introducing Equations 2 and 3 into Expression 1, after simple algebraic manipulations, E_C and E_P can be expressed as follows:

$$\begin{cases} E_C &= E\left[T\right] \cdot P_{ac} \\ E_P &= E\left[T_{ac}\right] \cdot \left(P_{ac} - P_{sl}\right) + E\left[T\right] \cdot P_{sl} \end{cases} , \tag{11}$$

where $E\left[T\right]$ and $E\left[T_{ac}\right]$ are given by Equations 5 and 10, respectively.



(a) CAM (b) PSM

Figure 5: Example of validation plots.

6 Evaluating the 802.11 Power Saving Mode

As mentioned in Section 3, the performance analysis of PSM is carried out by using both simulation and the analytical model derived in the previous section. Specifically, analytical results are used to provide better insights in the PSM behavior highlighted by simulation. As an example of the agreement between the analytical and the simulation model, Figure 5 shows two of the validation plots for E_C and E_P presented in [36].

According to the idle-time classification presented in Section 3, we analyze PSM performance by considering separately its behavior during interarrival times – i.e., during bursts (Section 6.2), and during User Think Times (Section 6.3).

6.1 Simulation Environment

Our simulator extends the model used in [16], and implements the reference environment described in Section 3. It simulates a full-compliant 802.11 hotspot (populated by a variable number of background mobile hosts), and full-compliant TCP-Reno between the mobile and the fixed host. Please note that the simulation model implements *all* the features of both 802.11 and TCP. To allow for significant values of burst sizes and User Think Times we make reference to the Web traffic. Therefore, each burst corresponds to the download of a Web page. In particular, we consider the statistical models of the Web traffic presented in the well-known works by Crovella et al. [13, 20].

A typical simulation run proceeds as follows (Table 1 summarizes the default values for the main simulation parameters). The tagged mobile host downloads N_{BR} bursts from the fixed server (recall that two consecutive bursts are spaced by a User Think Time). The average Web-page size and User Think Time duration (i.e., $E[d]$ and $E[UTT]$ in Table 1) are derived from [13, 20]. However, we tested the system over a wide range of burst and UTT values, making the analysis valid for the general traffic model presented in Section 3, and not only for the Web case. To mimic a realistic TCP connection between the mobile host and the fixed server, Internet Round Trip Times (as would be measured without PSM) are sampled from an exponential distribution (the default average value – RTT – is reported in Table 1). To simulate packet losses at Internet routers, TCP segments are randomly dropped with probability p_l^{tcp}. Note that p_l^{tcp} just accounts for losses in the wired network, due to routers' buffer overflow. The additional packet loss due to the WLAN depends on the MAC protocol behavior, and is thus not a simulation parameter (it can actually be derived by simulation). Finally, energy parameters are as follows. The power consumptions in the sleep and active modes (i.e., P_{sl} and P_{ac}) are the same as those used in [30]. These

Parameter	*Value*	*Unit*	*Parameter*	*Value*	*Unit*
N_{BR}	100	-	MSS	1460	B
$E[d]$	20.19	KB	P_{sl}	50	mW
$E[UTT]$	3.25	s	P_{ac}	750	mW
RTT	150	ms	t_{sa}	1	ms
p_l^{tcp}	1%	-	BI	100	ms

Table 1: Default simulation parameters

are quite similar to values used in other well-known analyses [23], and comparable to recent datasheets [18]. t_{sa} denotes the time required by the wireless interface to switch from the sleep to the active mode. Note that this parameter allows us to also include the cost of switching between the wireless interface operating modes according to PSM. Its default value is derived from the measurements in [30]. Specifically, [30] measured that, while operating in PSM mode, the wireless interface spends about 2 ms to switch from the sleep to the active mode and to receive a Beacon Frame. Based on the 802.11 standard [29] it is easy to show that the time required to receive a Beacon Frame is about 1 ms. Therefore, we assume 1 ms as the time required by the hardware to switch from the sleep to the active mode. We do not consider the impact of different t_{sa} values on PSM, because we focus more on networking and application-level parameters. Note that the switching time between operating modes (and thus the related costs) could be reduced significantly by improved hardware design (similarly to what is happening for channel switching in the mesh networks domain). Finally, the value of the Beacon Interval (BI) is the one suggested by the 802.11 standard [29]. To increase the results' reliability, each simulation experiment is replicated 10 times. Confidence intervals reported throughout the paper have 95% confidence level.

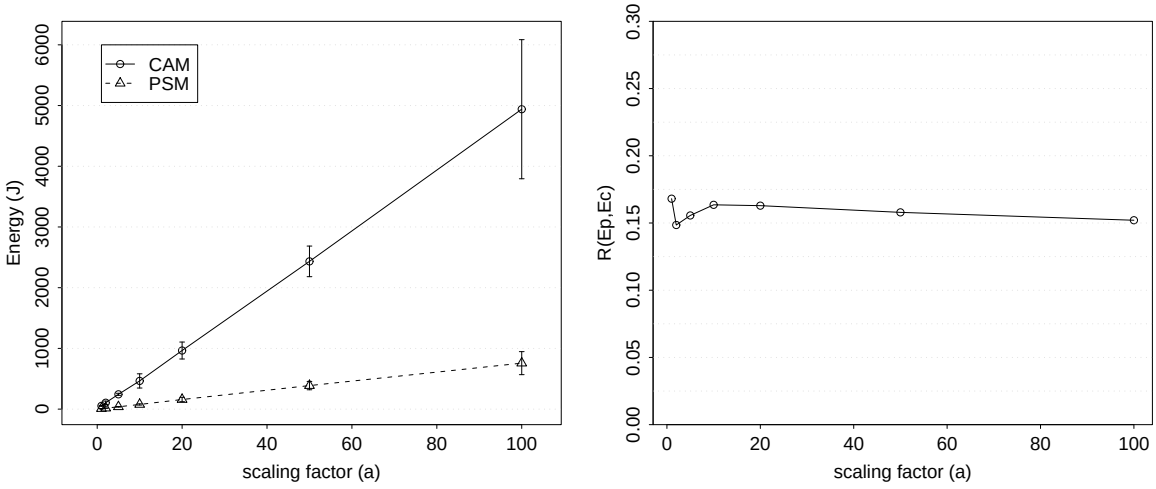
6.2 PSM performance during bursts

Bursts and interarrival times are determined by both application and networking protocols. In our scenario, where data mainly flow from the fixed server to the tagged mobile host, the application dictates the burst sizes[8], while the TCP protocol is the main responsible for interarrival times. Thus, we now focus on the impact of two parameters, i.e. i) the average burst size (Section 6.2.1), and ii) the TCP-connection throughput (Section 6.2.2). In both cases, we assume a single mobile host in the hotspot (i.e, $M = 0$). Section 6.2.3 extends the analysis by considering several mobile hosts in the same hotspot (i.e., $M > 0$).

6.2.1 Impact of the burst size

As mentioned above, in our model each burst corresponds to the entire download of a Web page. There is a wide consensus about the type of distribution for modeling page sizes (see, for example, [13, 20, 8]). On the other hand, the average value of this distribution can be highly variable, and can range from 20 KB up to few MB [13, 20, 21]. Based on these remarks, in our simulation model the burst-size distribution is defined by the random variable $a \cdot S$, where: i) a is a (integer) scaling factor, and ii) S is the random variable defining the page size distribution derived in [13, 20]. The average burst size can thus be scaled (by varying a) without modifying the distribution's coefficient of variation. This allows us to evaluate PSM under realistic traffic loads. Specifically, we report a set of experiments where a varies between 1 and 100, while the average of S (denoted by μ) is set to 20 KB [13, 20]. This results in an average burst size ranging from 20 KB to about 2 MB. We believe this range also represents the traffic when techniques such as loss-less compression or AJAX [25] are used, i.e. techniques that significantly reduce the amount

[8] E.g., in the Web case the burst sizes are determined by the content the user is downloading.



(a) Energy plots (b) $R(E_P, E_C)$

Figure 6: PSM performance as function of the average burst size ($a \cdot \mu$).

of data exchanged over the network per user request. More in general, we believe that this range represents all our reference applications.

As in this set of experiments we intend to investigate the PSM performance during bursts, User Think Times are always set to 0. Since the TCP-connection evolution depends on i) the average Round Trip Time (RTT) and ii) the segment-loss probability (p_l^{tcp}) [35], and both parameters can be reasonably assumed to be independent of the User Think Time duration, setting UTT to 0 is justified.

Figure 6(a) plots E_P (bottom curve) and E_C (top curve) for different average burst sizes. The most interesting feature is that energy increases linearly in both cases. This behavior can be explained by means of Equation 11. Since we assume $E[UTT] = 0$ and $E[d] = a \cdot \mu$, E_C becomes:

$$E_C = P_{ac} \cdot \frac{E[N_{BR}] \cdot a \cdot \mu}{\gamma_{TCP}} = a \cdot \mu \cdot K_C, \text{ where } K_C \triangleq P_{ac} \cdot \frac{E[N_{BR}]}{\gamma_{TCP}} . \quad (12)$$

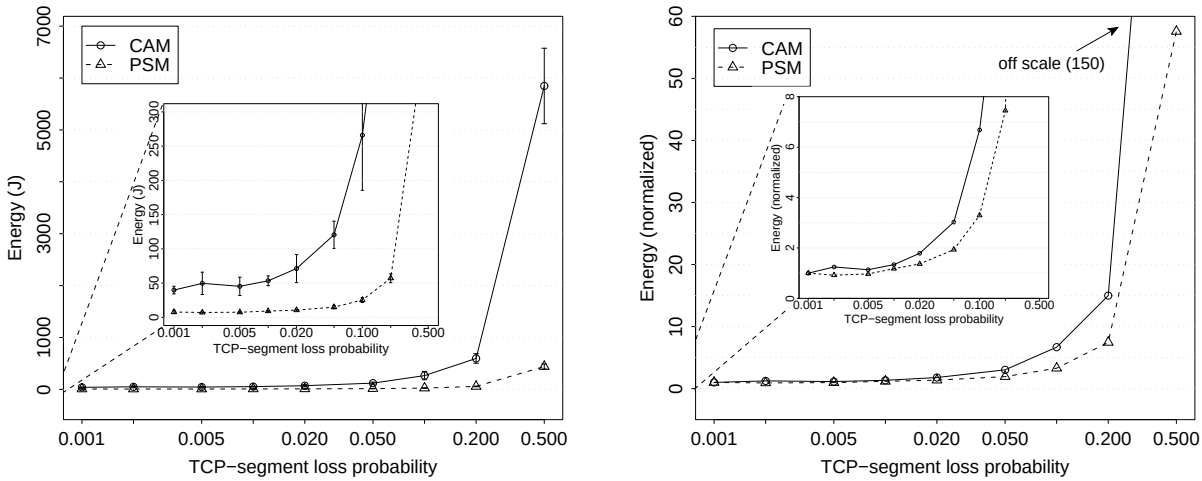
By following a similar line of reasoning, E_P can be expressed as follows:

$$E_P = a \cdot \mu \cdot K_P , \quad (13)$$

where K_P includes terms that are independent of both a and μ. Deriving the closed formula of K_P requires some manipulation . It can be expressed as $K_1 P_{sl} + K_3 (P_{ac} - P_{sl})$ where $K_3 = K_2 + K_1 \cdot b/BI$, $K_2 = (E[s] + E[a]) \cdot E[N_{BR}]/MSS$, and $K_1 = E[N_{BR}]/\gamma_{TCP}$ (see [2, 36]).

The linear increase of E_C and E_P with the average burst size has also an intuitive explanation. E_C is proportional to the the average download interval ($E[T]$, see Equation 11). Assuming $E[UTT] = 0$, the average download interval coincides with the average time spent downloading the bursts from the fixed host, i.e., $E[T_{data}]$. Furthermore, since the TCP throughput is assumed to be constant, $E[T_{data}]$ is proportional to the average burst size (see Equation 5).

In addition, E_P is a linearly increasing function of i) the average download interval ($E[T]$), and ii) the average time during which the tagged mobile host remains in the active mode (i.e., $E[T_{ac}]$, see Equation 11). Based on the above remarks, $E[T]$ is proportional to the average burst size. Now, we show that the same property holds for $E[T_{ac}]$, as well. $E[T_{ac}]$ includes two components, i.e., the time spent – within a download interval – to receive (transmit) TCP segments (TCP acks), and to receive Beacon frames from the Access Point (see Equation 10). The average total time required to receive (transmit) TCP segments (TCP acks) is proportional to the number of TCP segments (TCP acks) managed during the download interval, and, hence, to the burst size (Equation 8 and 10). The



(a) Energy plots

(b) Multiplicative factors

Figure 7: PSM performance as function of the TCP segment-loss probability (p_l^{tcp}).

total time required to receive Beacon frames is proportional to the number of Beacon Intervals within the download interval, thus to the download interval, and thus to the burst size.

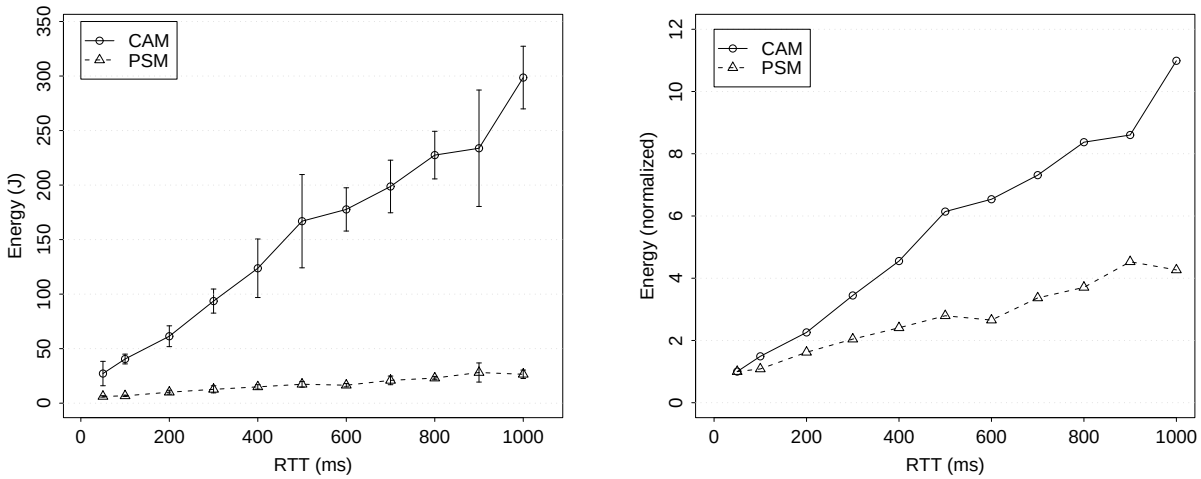
The results in Figure 6(a) highlight an important property of PSM, which is better emphasized in Figure 6(b). Figure 6(b) shows the $R(E_P, E_C)$ index[9] as a function of the a parameter. It clearly shows that $R(E_P, E_C)$ is almost independent of the average burst size. This is because E_P and E_C are both proportional to the burst size, and their ratio depends on the parameters that define K_P and K_C. In our experiments, this value is around 0.16, resulting in an energy saving of approximately 84%. Based on Figure 6(b) we can claim that *the energy saved by PSM does not significantly depend on the average burst size*. Therefore, in the following experiments, unless otherwise stated, we assume $a = 1$.

6.2.2 Impact of the Internet throughput

In this Section we investigate the impact on the PSM performance of the Internet throughput. The results presented in [35] show that the segment-loss probability (p_l^{tcp}) and the average Round Trip Time (RTT) are the main parameters that impact on the throughput of a TCP connection (γ_{TCP}). Specifically, γ_{TCP} is a decreasing function of both. Thus, we ran a set of simulation experiments to investigate the PSM behavior with respect to p_l^{tcp} and RTT, respectively.

According to [35], the lower and upper values of p_l^{tcp} are set to 0.001 and 0.5. $E[UTT]$ is set to 0, as above, while the rest of the simulation parameters are as in Table 1. Figure 7(a) plots E_P (bottom curve) and E_C (top curve) as functions of p_l^{tcp}. As expected, both E_P and E_C increase with p_l^{tcp}. It is well known that increasing p_l^{tcp} tremendously reduces the TCP throughput. The average duration of the download interval ($E[T]$) increases, and this results in an increase of both E_C and E_P. The additional download time mainly consists of longer idle times between burst segments. When PSM is not active, the additional time is spent completely in the active mode. When PSM is active, the additional time is only partly spent in the active mode (due to Beaconing), and mostly spent in the sleep mode. Hence, PSM is able to greatly reduce the negative effect of low throughput on the energy consumption. To quantify this behavior, let us focus on Figure 7(b) that shows the energy consumed at a given segment loss probability p_l^{tcp}, normalized to the energy consumed at $p_l^{tcp} = 0.001$. In a sense, Figure 7(b) shows, for each p_l^{tcp} value, the "energy multiplicative factor" with respect to the energy consumption at $p_l^{tcp} = 0.001$. For

[9] Recall that this index represents the fraction of energy spent when PSM is active, with respect to the energy spent when PSM is not active. Hence, it shows the energy saved by PSM.



(a) Energy plots (b) Multiplicative factors

Figure 8: PSM performance as function of the Round Trip Time (RTT).

example, when p_l^{tcp} is equal to 0.1, the multiplicative factor for E_C and E_P is around 7x and 3x, respectively. The multiplicative factor when PSM is active is always lower than when it is not. Furthermore, the more p_l^{tcp} increases, the more the difference between the two curves increases.

A similar result is also obtained when analyzing the dependence of the energy consumption on RTT (Figure 8(a,b)). Though the absolute values are different from those in Figures 7(a) and 7(b), the qualitative behavior is the same. Hence, we can conclude that the energy consumption is negatively affected by low TCP throughput, either PSM is active or not. However, PSM greatly helps in mitigating this effect.

6.2.3 Impact of the WLAN contention

So far, the analysis has been carried out under the assumption of a single mobile host within the Wi-Fi hotspot, i.e., M has been assumed to be equal to 0. Now, we evaluate the impact of MAC-level contention on the mobile-host energy consumption (i.e., $M > 0$). To this end, we firstly highlight the limitations of PSM when a standard TCP architecture is used. Then, we investigate up to what extent an Indirect-TCP architecture [11] can alleviate these problems. The simulation parameters are as shown in Table 1, apart from $E[UTT]$ which is set to 0, as above.

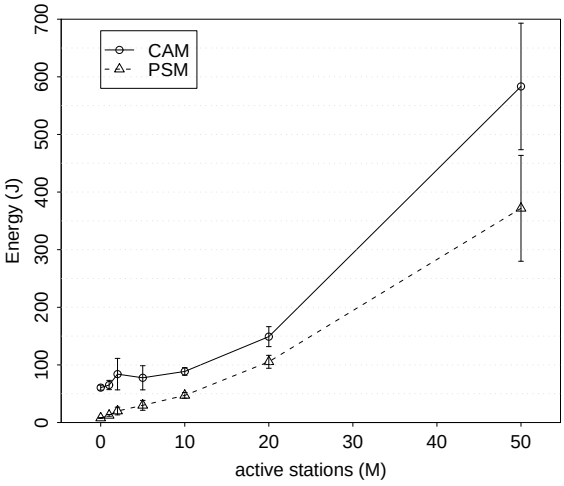
802.11 PSM in a standard TCP architecture

Figure 9(a) plots E_P and E_C As expected, both E_P and E_C increase when the contention in the WLAN increases. This behavior stems from two causes: on one hand, MAC-level contention reduces the TCP throughput; on the other hand, it increases the MAC delay.

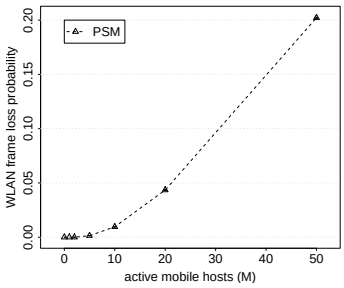
The impact of the WLAN contention on the TCP throughput clearly appears from Figures 9(b,c,d). The frame loss probability on the WLAN increases with M (Figure 9(b)). This results in increased number of timeouts at the TCP sender (Figure 9(c)), and, ultimately, to a severe degradation of the TCP throughput (Figure 9(d)).

In addition, it is well known that increasing the MAC-level contention increases the MAC delay [15]. When the MAC delay increases, the time required for receiving a TCP segment (i.e., $E[s]$ in Equation 10), or sending a TCP ack (i.e., $E[a]$ in Equation 10), increases accordingly. So, the time interval during which the tagged mobile host is active ($E[T_{ac}]$), and hence E_P, increases with M (see Equations 10 and 11). Clearly, similar remarks apply to E_C as well.

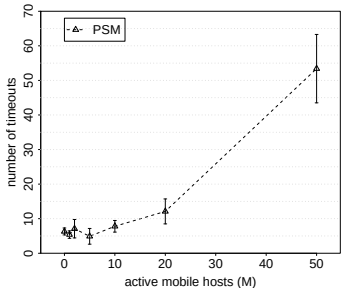
Based on these remarks, two factors are responsible for the increased energy consumption when M increases, i.e., i) the reduced TCP throughput (due to an increase in the frame loss probability); and ii) the increased MAC



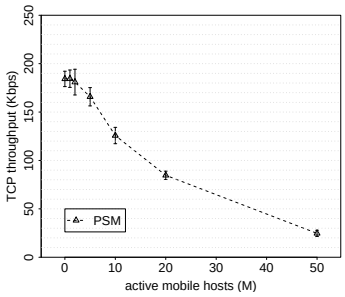
(a) Energy expenditure



(b) WLAN frame loss probability



(c) number of TCP timeouts



(d) TCP throughput

Figure 9: 802.11 PSM performance in a standard TCP architecture.

delay. In the following of this section, we decouple the effects of these two factors, to understand the real impact of each one. Specifically, we show that using an Indirect-TCP architecture [11] eliminates factor i), and explains the discrepancy between the E_P and E_C curves in Figure 9(a).

802.11 PSM in an Indirect TCP architecture

In an Indirect-TCP architecture [11], the transport connection between the mobile host and the fixed host is split in two distinct parts at the boundary between the wireless and the wired networks (i.e., at the Access Point). An agent (the Indirect-TCP Daemon) relays the data between the two parts of the connection granting transparency to the application level. It has been proved [12] that this architecture shields the TCP sender at the fixed host from the losses on the wireless link, thus increasing the throughput with respect to the legacy TCP architecture.

We show that this "shielding property" can be exploited to eliminate the energy wastage related to the transport

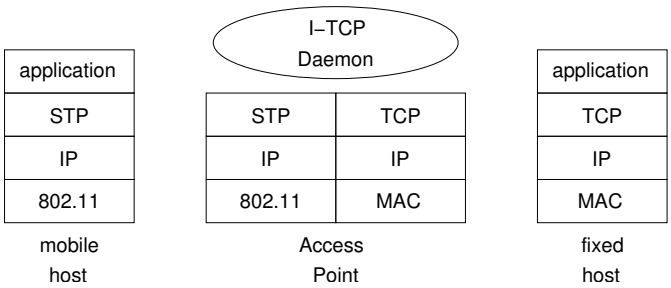
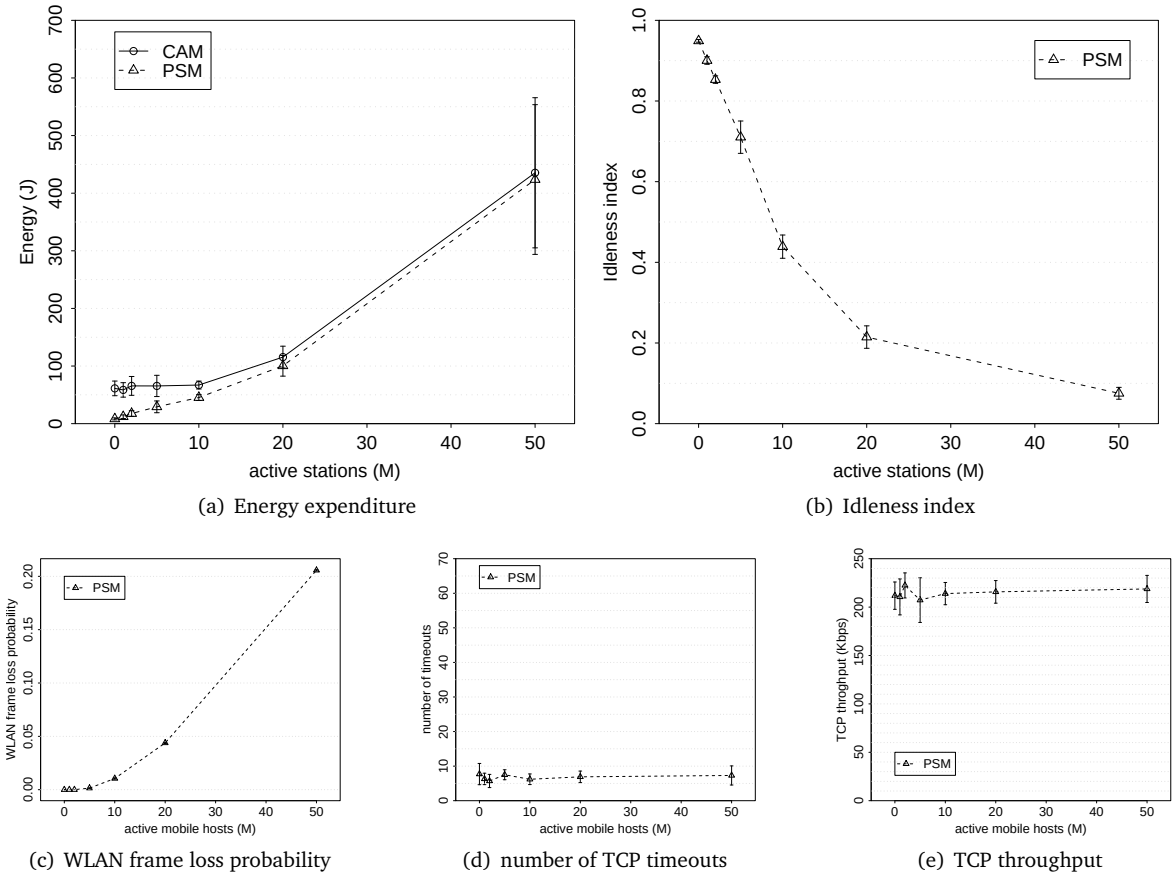


Figure 10: Indirect-TCP architecture



(a) Energy expenditure (b) Idleness index

(c) WLAN frame loss probability (d) number of TCP timeouts (e) TCP throughput

Figure 11: 802.11 PSM performance in an Indirect-TCP architecture.

protocol (i.e., cause i) above). To this end, we run simulations by replacing the standard TCP architecture with the architecture shown in Figure 10. This is similar to the original Indirect TCP, except for the transport protocol used over the WLAN. Specifically, we use the Simplified Transport Protocol (STP), which is essentially a Stop-and-Wait transport protocol, optimized for the one-hop wireless environment [6, 3].

Figures 11(c,d,e) show that the Indirect-TCP architecture actually shields the TCP sender at the fixed host from frame losses in the WLAN (note that the TCP throughput is measured at the fixed host). Specifically, even though the WLAN frame loss probability increases just as in the legacy TCP architecture (compare Figures 11(c) and 9(b)), the number of timeouts registered at the TCP sender (Figure 11(d)) and the throughput experienced by the TCP connection over the wired network (Figure 11(e)) are independent of that. Hence, the effect of the reduced TCP throughput on the energy consumption, registered in the previous set of experiments, disappears. Only the MAC-delay increase (cause ii) above) is thus responsible for the additional energy consumption. It should be noted that PSM is not able to face this problem, as it appears from Figures 11(a,b). Figure 11(b) shows the Idleness index as a function of M. The Idleness index is defined as the fraction of time (within bursts) during which the tagged mobile host is idle because there are no frames buffered for it at the Access Point. When the WLAN contention is high ($M = 50$) the transport-level throughput on the WLAN is lower than the TCP-throughput on the wired part of the connection. Hence, the TCP sender pumps data towards the Access Point at a higher rate than the tagged mobile host could fetch from the Access Point. So, the tagged mobile host is never idle, and the PSM can never switch the wireless interface to the sleep mode. In conclusion, for high contention levels, either enabling the PSM or not leads to similar results (Figure 11(a)). Based on these observations, we can conclude that the effect of the MAC delay on the energy consumption can be contrasted only by reducing the MAC delay itself through MAC-level modifications

(e.g., as proposed in In [14]).

To summarize, the results presented so far show that in our reference scenario PSM works very well during bursts, i.e., *it manages interarrival times very effectively*. Specifically, we have shown that: i) the energy saving achieved by PSM is almost independent of the size of bursts that are downloaded, and, for typical values of the main Internet parameters, it can be as high as 84%; and ii) PSM is able to limit the energy consumption when the throughput offered by the TCP connection drops.

6.3 Is PSM effective with any class of idle times?

Since PSM just exploits the sleep mode of the wireless interface, one could argue that it could be improved by using the off mode instead. However, this would cost additional delay and energy consumption upon re-activation. While the transition time from the sleep to the active mode (t_{sa}) is in the order of 1 ms, the transition time from the off to the active mode (throughout referred to as t_{oa}) is quite greater. The work in [1] measured a transition time around 400 ms, while [43] measured a transition time around 100 ms[10], which is the value we use hereafter. As highlighted in Section 6.1, we choose not to focus on the impact of different switching times on energy consumption. Intuitively, the sleep mode should be more appealing for "short" idle times, while for "long" idle times the best choice should be switching the wireless interface off. In this section we corroborate this claim by means of the analytical model introduced in Section 5. This suggests some directions to improve the standard PSM.

Let us focus on an idle time of a given length (say, t_i), and let us define the behavior of two ideal energy managers, just exploiting the sleep and the off mode, respectively. In the ideal case, these energy managers know a-priori the length of the idle time. The energy manager that uses the sleep mode keeps the wireless interface sleeping up to t_{sa} seconds before the idle-time endpoint. If $E_S(t_i)$ denotes the energy spent by this energy manager during t_i, the following equation holds:

$$E_S(t_i) = (t_i - t_{sa}) \cdot P_{sl} + t_{sa} \cdot P_{ac} = t_i \cdot P_{sl} + (P_{ac} - P_{sl}) \cdot t_{sa} \ . \tag{14}$$

On the other hand, the ideal energy manager that uses the off mode lets the wireless interface in the active mode if t_i is less than t_{oa}. Otherwise, it switches it off, and reactivates it t_{oa} seconds before the idle-time endpoint. If $E_O(t_i)$ denotes the energy spent in this case, the following equation holds:

$$E_O(t_i) = \begin{cases} t_i \cdot P_{ac} & \text{if } t_i \leq t_{oa} \\ t_{oa} \cdot P_{ac} & \text{otherwise} \end{cases} \ . \tag{15}$$

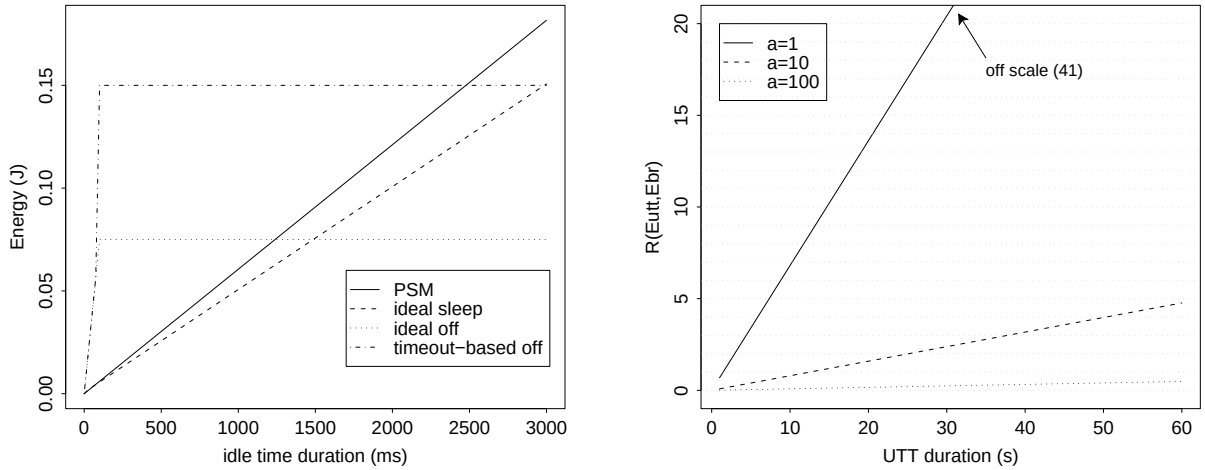
Figure 12(a) plots Equations 14 ("ideal sleep" curve) and 15 ("ideal off" curve) as functions of t_i. It confirms that for "short" idle times the best policy consists in putting the wireless interface in the sleep mode, while for "long" idle times the off-based policy exhibits the best performance. Let $\hat{t}_i$ denote the crossing point between $E_S(t_i)$ and $E_O(t_i)$. Then, the optimal (ideal) policy is a *mixed policy* that uses the sleep mode for idle times lower than $\hat{t}_i$, and the off mode for idle times greater than $\hat{t}_i$. This analysis also suggests that mixed policies using both the sleep and the off modes should be defined when "short" and "long" idle times coexist, as in the case of our reference applications.

Let us now analyze the energy spent by PSM during t_i. Since the station is active just to receive Beacons, the average energy spent by PSM during t_i ($E_P(t_i)$) is:

$$E_P(t_i) = \left[t_i - \frac{t_i}{BI} \cdot b \right] \cdot P_{sl} + \frac{t_i}{BI} \cdot b \cdot P_{ac} = t_i \cdot \left[P_{sl} + (P_{ac} - P_{sl}) \cdot \frac{b}{BI} \right] \ . \tag{16}$$

Equation 16 is plotted in Figure 12(a), with label "PSM". This plot confirms that PSM is effective with respect to interarrival times, i.e., for idle times below 1 s. The additional energy expenditure achieved by PSM with respect

[10] Actually, 100 ms is the time measured for a complete cycle active-off-active. Since in our analysis the breakdown between the active-off and off-active times is not important, we assume 100 ms as the off-active transition time.



(a) off- and sleep-based strategies

(b) Relative cost of UTTs and bursts in PSM

Figure 12: Evaluation of PSM during User Think Times.

to the "ideal-sleep" policy is always below 20%. Thus, in this region, PSM is a *close approximation of the best, ideal, policy*.

Figure 12(a) also shows that the PSM discrepancy with off-based policies increases as idle times become longer and longer. The "ideal-off" policy cannot be implemented in practice. However, let us consider a very simple timeout-based policy that lets the mobile host active for the first t_{oa} seconds of an idle time, and then switches it off[11]. The energy spent by this policy is plotted in Figure 12(a) for comparison ("timeout-based off" label). This policy is known to be *2-competitive*, i.e., it never consumes more than twice the energy spent by the ideal off-based policy [26]. Though this policy can be significantly improved [26, 3], it performs better than PSM for idle times longer than 2.5 s, and even better than the "ideal-sleep" policy for idle times longer than 3 s. Therefore, designing a mixed policy that exploits the off mode during long idle times and PSM during short idle times is an interesting direction to pursue.

Before analyzing in detail how such improvements can be implemented in a feasible way, let us further investigate how much energy is spent by PSM during User Think Times, with respect to the energy spent during bursts. This indicates if it is actually worth designing a system that reduces the energy spent during User Think Times. Let us define E_{BR} as the average energy spent by PSM to download a single burst, and E_{UTT} as the average energy spent by PSM during a User Think Time[12]. In Figure 12(b) the index $R(E_{UTT}, E_{BR})$ is plotted for increasing User Think Times. Three different plots are drawn for three different average burst sizes, i.e., $a = 1$, $a = 10$, and $a = 100$. Figure 12(b) shows that the energy spent during User Think Times is not negligible with respect to the energy spent during bursts, for any average burst size. Specifically, for small bursts (i.e., $a = 1$), $R(E_{UTT}, E_{BR})$ is around 20 for UTTs equal to 30 s, and raises up to about 40 for UTTs equal to 60 s (not shown in the plot). Even for large bursts (i.e., $a = 100$), the energy spent during the User Think Times is about 25% for UTTs equal to 30 s, and about 50% for UTTs equal to 60 s. This is a strong motivation to look for possible improvements of PSM in the region of *long* idle times.

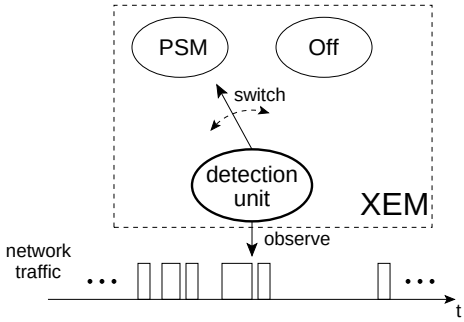


Figure 13: Cross-Layer Energy Manager: a conceptual scheme

7 Enhancing PSM: a Cross-Layer Approach

The *Cross-Layer Energy Manager* (XEM) implements a mixed policy. A conceptual scheme of XEM is shown in Figure 13. XEM observes the traffic generated by the tagged mobile host, and switches the wireless interface between PSM and off mode accordingly. Thus, XEM includes a *detection unit* that implements two *detection algorithms* for detecting the beginning of bursts and User Think Times, respectively. Unlike PSM (and many of its modifications referred in Section 2) XEM does not work exclusively at the MAC level. Instead, it exploits information related to different layers in the protocol stack, and thus leverages the powerful cross-layer approach [19].

In the following of this section we define some possible detection algorithms, and evaluate the corresponding XEM implementations. Although very simple, these implementations are very effective solutions. Furthermore, it should be noted that, thanks to its flexible design, XEM is able to accommodate also different (possibly more sophisticated and even more effective) detection algorithms and energy-saving policies. For example, during bursts it would be possible to use BSD, STPM or SPSM, instead of PSM.

7.1 Burst detection

In a Wi-Fi hotspot, detecting the beginning of a burst is usually not a big deal. The main applications that are suitable to be deployed in Wi-Fi hotspots (e.g., Web, mail, file download) follow a client/server paradigm, the mobile host acting as the client. Thus, bursts represent data that are downloaded after the mobile host has sent a request to the fixed host. In other words, it is reasonable to assume that *the first segment of a burst is sent by the mobile host*. Under this assumption, the beginning of a burst can be easily detected at the mobile host, and identified by the request sent by the client application (typically after a User Think Time). Therefore, XEM simply lets the mobile host in the off mode during User Think Times, and switches it to the standard PSM as soon as a new application-level request is detected. Section 7.4 discusses how to extend XEM to more general scenarios.

7.2 User Think Time Detection

User Think Time detection could exploit knowledge about the application(s) behavior. For example, [6] presents two different energy-management policies designed to support Web-based applications[13] and implemented at the middleware layer. They both rely on an agent at the mobile host that spoofs the Web traffic generated by the user. For each Web page, this agent is aware of the set of files composing the page itself. Once all of these files have been downloaded, a User Think Time is assumed to start. This allows detecting User Think Times as soon as they start.

[11] This policy is feasible if one supposes that the mobile host is immediately aware of the availability of the first segment next to the idle time. We discuss this point in Section 7.

[12] E_{BR} can be easily computed from the analytical results provided in Section 5, while E_{UTT} is equal to E_P ($E[UTT]$).

[13] The reference environment is similar to the one considered in this paper.

```
1: while true do
2:    net_interface_mode = PSM
3:    collect the list of files composing the Web page
4:    repeat
5:       spoof the application-level traffic
6:    until the whole Web page is at the mobile host
7:    net_interface_mode = off
8:    wait(Web-page request from the application)
9: end while
```

```
1: while true do
2:    net_interface_mode = PSM
3:    is_UTT = false
4:    repeat
5:       wait(beginning of idle time)
6:       it_end = false
7:       repeat
8:          t= update the idle-time duration
9:          if t ≥ t_TO then
10:            is_UTT = true
11:         else if a new segment is either sent or received then
12:            it_end = true
13:         end if
14:      until it_end==true or is_UTT==true
15:   until is_UTT==true
16:   net_interface_mode = off
17:   wait(packet from the application)
18: end while
```

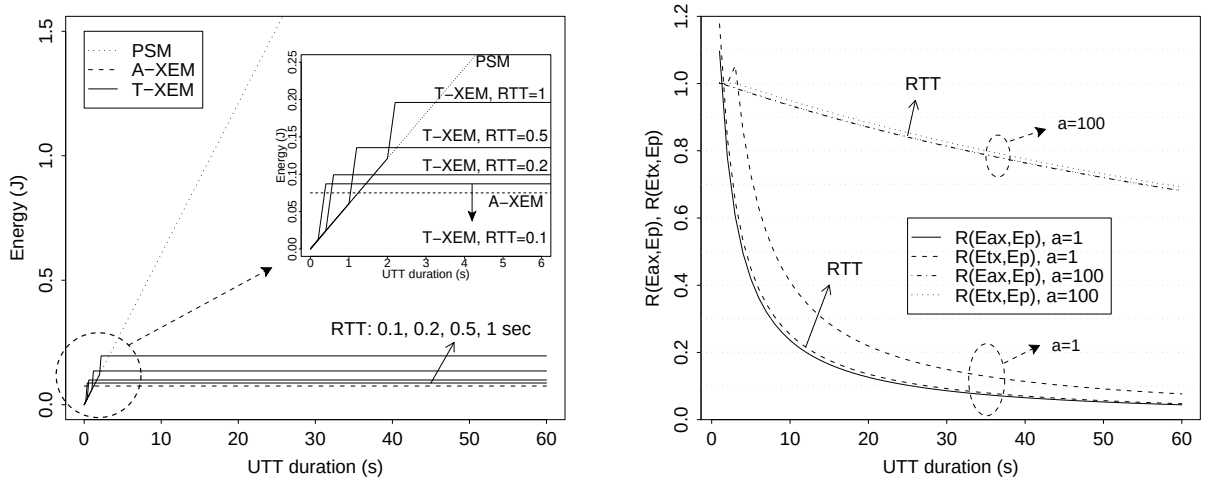
Figure 14: Cross-Layer Energy Managers: A-XEM (left) and T-XEM (right)

A first way for detecting User Think Times in XEM is inspired by this approach. This version of XEM includes a middleware agent that is aware of the specific application running on the mobile host (e.g., Web browsing). As this implementation of the Cross-Layer Energy Manager depends on the specific application it is designed for, it is hereafter referred to as the *Application-dependent* Cross-Layer Energy Manager (A-XEM). The pseudo-code specification of this Energy Manager is shown in Figure 14(left) (Web browsing is used as the reference application). Let us focus on line 2, and assume that a burst just started. According to the general XEM scheme depicted in Figure 13, A-XEM relies upon the standard PSM during bursts (lines 2-6), and switches the wireless interface off during User Think Times (lines 7-8). The completion of a Web-page download triggers the start of a User Think Time (line 6), while a new request from the user indicates that a new burst is starting (line 8). Please note that, apart from the PSM functionalities already included in the Access Point, A-XEM can be entirely implemented at the mobile host.

A-XEM uses an off-based policy to manage User Think Times, which may be suboptimal for very short UTTs. As it is shown in Section 7.3, the penalty paid for this – in terms of energy consumption – is very limited. Moreover, additional mechanisms should be included to improve A-XEM performance during short User Think Times. In this paper we decide not to explore this direction in order to keep the A-XEM definition simple.

A-XEM is strictly tied with the application it is designed for. Hence, a customized energy manager is needed for each network application. Furthermore, a coordination between different energy managers is needed in presence of concurrent applications. These drawbacks can be overcome, at the cost of a little performance degradation, by implementing an application-independent Cross-Layer Energy Manager. In the following we define a Cross-Layer Energy Manager that relies upon a timeout-based policy to detect User Think Times. Hence, this energy manager is referred to as the *Timeout-based* Cross-Layer Energy Manager (T-XEM). In [3] it is shown that, in our (TCP) environment, interarrival times can be thought of as time intervals between consecutive TCP segments. Due to the TCP behavior, new TCP segments are expected (at worst) one RTT after a TCP ack has been sent by the mobile host. If no TCP segments have arrived after one RTT, it is reasonable to assume that a UTT has started. Thus, T-XEM derives, on-line, a statistical characterization of the RTT between the mobile and the fixed host. Based on this characterization, a timeout value (denoted by t_{TO}) is chosen, in such a way that idle times longer than t_{TO} are, very likely, User Think Times. In other words, if at some point in time an idle time is detected, and t_i is the time elapsed from its beginning, the equation $p\,(t_i \text{ is a UTT} | t_i \geq t_{TO}) = 1$ is assumed to hold.

The pseudo-code specification of T-XEM is detailed in Figure 14(right). As in the case of A-XEM, T-XEM is implemented at the mobile host (apart from the PSM functionalities already implemented at the Access Point). Let us focus on line 2, and assume that a burst just started. T-XEM switches the mobile-host wireless interface to PSM,



(a) Energy consumption just during UTTs

(b) Energy consumption during a single cycle

Figure 15: Evaluation of XEM.

and executes lines 4-15 while the burst is ongoing. T-XEM waits for the beginning of an idle time (line 5) and, then, monitors its duration (line 8). One of the following conditions may occur, i.e.: i) the idle time is longer than t_{TO} (lines 9-10); or ii) a new TCP segment is received, or a new TCP ack becomes ready for transmission (lines 11-12). In case ii) the detected idle time is clearly an interarrival time inside the ongoing burst. Therefore, T-XEM skips to line 5 and waits for the next idle time. In case i) (i.e, when a User Think Time is detected), T-XEM switches the wireless interface off (line 16). The wireless interface remains off until a new burst is detected, i.e., until a new request is generated by the application at the mobile host (line 17). At this point in time, T-XEM switches the wireless interface to PSM (line 2), and waits for the next idle time, as explained above (lines 4-15).

7.3 XEM Performance Evaluation

In this Section we exploit the analytical model derived in Section 5 to evaluate the improvements, in terms of energy saving, achieved by the Cross-Layer Energy Managers (A-XEM and T-XEM) with respect to the standard PSM. As far as T-XEM, the timeout value (i.e., t_{TO}) is defined as $t_{TO} \triangleq 2 \cdot RTT$, where RTT denotes the (sampled) average value of the Round Trip Time (RTT). The assumption behind this choice is that the probability of sampling a Round Trip Time longer than twice the average value is negligible. Four RTT values are considered in the following analysis, i.e., RTT=100 ms, 200 ms, 500 ms and 1 s, respectively.

A first set of experiments is aimed at evaluating the sensitiveness of the Cross-Layer Energy Managers with respect to the User Think Time duration. Figure 15(a) shows the energy consumption of T-XEM and A-XEM for increasing UTTs (for T-XEM, a different curve is plotted for each RTT value). The energy consumption of PSM, derived from derived from Equation 16, is also shown for comparison. The energy consumption of A-XEM (E_{AX}) is constant, and equal to $t_{oa} \cdot P_{ac}$. Finally, the energy consumption of T-XEM is as follows:

$$E_{TX}(UTT) = \begin{cases} E_P(UTT) & UTT \leq t_{TO} \\ E_P(t_{TO}) + t_{oa} \cdot P_{ac} & UTT > t_{TO} \end{cases}, \quad (17)$$

where $E_P(\cdot)$ is the PSM energy consumption. Equation 17 can be explained by recalling that the T-XEM lets the wireless interface in PSM for User Think Times shorter than t_{TO}. Thus, in this range, the energy consumption of PSM and T-XEM is exactly the same, i.e. $E_P(UTT)$. On the other hand, for User Think Times greater than t_{TO}, T-XEM lets the wireless interface in PSM for the first t_{TO} seconds, and, then, switches it to the off mode.

As anticipated above, both A-XEM and T-XEM perform worse than PSM for very short User Think Times. How-

ever, the region where this occurs is limited to very small User Think Times, in the order of few seconds. As highlighted in Section 3, the probability of having such small UTTs is very low. Figure 15(a) shows that, for typical UTT values (tens of seconds), the Cross-Layer Energy Managers greatly outperform PSM. It is also interesting to compare the performance of the two XEM implementations. Clearly, A-XEM exhibits the best performance. T-XEM consumes 1.2 to 2.6 times the energy spent by A-XEM (for RTT equal to 0.1 s, and 1 s, respectively).

To complete the analysis we now consider the energy consumed by the Cross-Layer Energy Managers not only during User Think Times, but also within bursts. To this end, we assume that during bursts T-XEM never detects false User Think Times, i.e., we assume $p\left(t_i \text{ is a UTT} | t_i \geq t_{TO}\right) = 1$. Under this assumption, T-XEM behaves exactly as PSM during bursts (please note that the same property also holds for A-XEM). By exploiting the analytical formulations of E_P, E_{AX} and E_{TX}, we can evaluate the energy spent by PSM, A-XEM and T-XEM, respectively, during a single burst followed by a User Think Time. These quantities are throughout referred to as $E_P^{(1)}$, $E_{AX}^{(1)}$ and $E_{TX}^{(1)}$. Accordingly, indexes $R(E_{AX}^{(1)}, E_P^{(1)})$ and $R(E_{TX}^{(1)}, E_P^{(1)})$ are evaluated and plotted in Figure 15(b) for increasing UTTs. In Figure 15(b) we only consider the lower and upper value for T-XEM, i.e., 0.1 s and 1 s. Furthermore, to investigate the performance of the Cross-Layer Energy Managers for a wide range of burst sizes, we considered both short (i.e., $a = 1$) and long (i.e., $a = 100$) burst sizes.

For typical User Think Times, the improvement over PSM is quite evident. For example, for short burst sizes (i.e., a=1), and a User Think Time of 30 s, A-XEM spends *just* 8.6% of the energy consumed by PSM, while T-XEM spends always *less* than 15% of the energy consumed by PSM. These values *drop further* to 4.4% and 7.7%, respectively, when the User Think Time increases to 60 s. As expected, the performance gains are reduced if we focus on a particular User Think Time, and increase the burst sizes (e.g., set a to 100). This is because, for a given User Think Time, the (energetic) cost of a burst (with respect to the cost of the User Think Time) increases with the burst size (see Figure 12(b)). Since A-XEM and T-XEM differ from PSM in the way they handle User Think Times, the energy saved with respect to PSM is reduced when the burst size increases. In detail, for User Think Times equal to 30 s and 60 s, the energy saved by the Cross-Layer Energy Managers with respect to PSM is about 20% and 30%, respectively. It is also interesting to note that, as the average burst size increases, the performance difference between A-XEM and T-XEM becomes almost negligible, since for large bursts the energy consumed during bursts dominates the energy consumed during UTTs. This implies that for large bursts is not so important to consider very sophisticated algorithms to detect User Think Times.

In conclusion, the above analysis has shown that Cross-Layer Energy Managers exhibit significant improvements, in terms of energy saving, with respect to PSM. For typical values of the User Think Time (i.e, 30 s), the additional energy saving is at least 20% (for large bursts), and can be as high as 91% (for small burst sizes). For larger UTTs (i.e., 60 s) the additional energy saving is at least 30%, and can be as high as 96%.

To conclude the XEM analysis, we qualitatively compare it with STPM, BSD and SPSM. To this end, it is worth recalling the index id, defined in Section 4 as the ratio between the energy saved by PSM, and the energy saved by an ideal policy, that completely eliminates the energy consumption due to the wireless interface. id can be expressed as $id = 1 - \beta - \frac{d}{f}$ where β is the energy saved by PSM (relative to the wireless interface), d is the additional delay introduced by PSM, and f is the ratio between the power consumption of the wireless interface and the rest of the device. If we focus on the download of a single page, followed by a UTT, d becomes the additional delay introduced by PSM on the complete cycle. Since the UTT length is usually quite larger than the download time, it is reasonable to assume $d = 0$ also in the PSM case (we already discussed in Section 4 that $d = 0$ applies also to the best cases of STPM, BSD and SPSM). d can be set to 0 also in the case of XEM, since it introduces just 100 ms to the PSM additional delay. Therefore, the difference between XEM and the other techniques relies in the different values of β they are able to achieve. Figure 16 shows the range of XEM performance presented in Figure 15(b), and the maximum expected performance of PSM, STPM, BSD and SPSM. Specifically, i) STPM behaves exactly like PSM during a UTT; ii) BSD in the best case listens for a Beacon just every 900 ms, and sleeps for the rest of the time; iii)

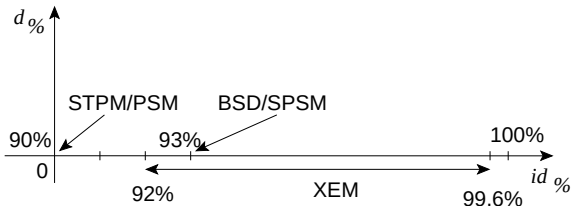


Figure 16: Comparison between XEM, PSM, STPM, BSD and SPSM.

SPSM may be able to sleep for the whole UTT. XEM achieves higher energy saving because, unlike these policies, it exploits the off mode of the wireless interface during UTTs. Authors of [30] actually envision a similar BSD extension, but do not analyze it in detail. Authors of [1] define a STPM+ policy that is able to exploit also the off mode, but do not analyze it during UTTs. Anyway, XEM does not require MAC-level modifications (unlike BSD), and we believe that it is a simpler solution with respect to STPM+.

7.4 Relaxing XEM assumptions

In the definition of XEM we have assumed that i) a single network application is running at the mobile host; ii) this application does not open parallel TCP connections with the server; and iii) this application acts as a client, i.e., new bursts start with a request sent by the mobile host to the (fixed) server. All of these assumptions were aimed at simplifying the XEM definition and analysis. However, they can be easily relaxed with simple modifications to XEM. Let us start by focusing on assumption ii). A-XEM can be used unchanged even in the case of multiple TCP connections. Actually, A-XEM detects User Think Times and new bursts generated by the application irrespectively of the number of TCP connection used. T-XEM could monitor the traffic exchanged between the mobile host and the Access Point, irrespectively of the particular TCP connections, to detect User Think Times. A User Think Time would be detected when the the mobile and fixed hosts do not exchange any data for t_{TO} seconds. Good candidate values for t_{TO} would be calculated on the basis of self-learning algorithms, which have shown to be able to estimate the statistical features of the joint traffic produced by concurrent applications using parallel TCP connections (see [5] for details). It can be easily shown that this T-XEM modification would allow us to relax assumption i) as well. Assumption i) could be relaxed also for A-XEM. Specifically, in the case of concurrent applications several instances of the application-specific detection algorithms defined by A-XEM would be concurrently operating on the mobile host. A further A-XEM module, i.e., a coordination module, would coordinate detections related to each specific application, and would be responsible for switching the wireless interface of the mobile host between the PSM and off mode.

Finally, XEM can be extended to relax the third assumption as well, and support mobile hosts acting as servers (i.e., able to receive asynchronous requests from the Internet). To this end, XEM would periodically switch the wireless interface of the mobile host to PSM during User Think Times. This way, frames that could have been buffered at the Access Point would be downloaded by exploiting the PSM mechanisms. Furthermore, XEM would switch again the wireless interface off if no new data are exchanged for a Beacon Interval. Obviously, this XEM extension would have some additional energetic cost, since more switching-on events are required, and more time would be spent by the wireless interface in PSM.

8 Summary and Conclusions

In this paper we have extensively evaluated the performance of the 802.11 PSM in terms of energy consumption as a function of a number of application and network parameters, and as a function of the MAC-level congestion.

The main results can be summarized as follows. During traffic bursts PSM is quite effective, and able to save up to 90% of the energy spent without energy management. It works remarkably well for a wide range of of burst sizes. Furthermore, it is able to significantly reduce the negative effect on energy consumption of low transport-level throughput and MAC-level contention. Unfortunately, PSM is not very fit to deal with User Think Times between bursts. We have shown that this originates from the fact that PSM switches the wireless interface to the sleep mode during any type of idle time. During long idle times, such as UTTs, switching it to the off mode proves to be more energy efficient. Therefore, we have proposed and evaluated XEM, a Cross-Layer Energy Manager that uses PSM during bursts, and switches the wireless interface off during UTTs. XEM implements very simple yet efficient algorithms to detect the beginning of bursts and UTTs, without requiring any modification to legacy-Internet applications or to the standard 802.11. XEM is able to achieve energy saving between 20% and 96% with respect to the standard PSM.

Our opinion is that these improvements stem from the cross-layer nature of the XEM design. Specifically, PSM just uses MAC-level information (i.e., availability of frames to/from the mobile host) to detect idle times, and manages the mobile host's wireless interface accordingly. By operating exclusively at the MAC level, PSM is not flexible enough to cope with the network traffic generated by typical Internet applications in Wi-Fi hotspots. In particular, PSM is not able to distinguish between short idle times (within bursts) and long idle times (between consecutive bursts), and is thus not able to dynamically select the best energy-saving policy. On the other hand, XEM dynamically chooses between sleep-based and off-based policies, according to the type of idle time that is occurring. Furthermore, the algorithms it uses to detect idle times (and distinguish between different idle-time types), exploit information residing at different layers in the protocol stack, from the MAC up to the application layer. The performance improvements presented in this paper show that such a cross-layer approach is very promising.

The main contribution of this paper is thus twofold. To the best of our knowledge, this is the first work in the literature that provides such a comprehensive understanding of PSM strengths and weaknesses in terms of energy saving. A further contribution is showing that a cross-layer design is a very suitable direction to deal with the energy-management problem in WLANs, by enhancing PSM in the cases where it is not efficient.

References

[1] M. Anand, E. Nightingale, and J. Flinn, "Self-Tuning Wireless Network Power Management", Wireless Networks, Vol. 11, Issue 4, pp. 451-469, Jul. 2005.

[2] G. Anastasi, M. Conti, E. Gregori and A. Passarella, "802.11 Power-Saving Mode in Wi-Fi hotspots: Limitations, Enhancements and Open Issues", Information Engineering Department, University of Pisa, Tech. Rep. DII-0012-2005, available on-line at http://www.ing.unipi.it/~o783499/research/docs/PScross-tr.pdf.

[3] G. Anastasi, M. Conti, E. Gregori and A. Passarella, "A Performance Study of Power-Saving Policies for Wi-Fi Hotspots", Computer Networks, Vol. 45, Issue 3, pp. 295-318, June 2004.

[4] G. Anastasi, M. Conti, E. Gregori and A. Passarella, "Saving Energy in Wi-Fi Hotspots through 802.11 PSM: an Analytical Model", Proceedings of the Second Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt 2004), Cambridge (UK), March 24-26, 2004.

[5] G. Anastasi, M. Conti, E. Gregori and A. Passarella, "Experimental Analysis of an Application-independent Energy Management Scheme for Wi-Fi Hotspots", Proceedings of the Ninth IEEE Symposium on Computers and Communications (ISCC 2004), Alexandria (Egypt), June 29 -July 1, 2004.

[6] G. Anastasi, M. Conti, E. Gregori and A. Passarella, "Performance Comparison of Power Saving Strategies for Mobile Web Access", Performance Evaluation, Vol. 53, N. 3-4, August 2003, pp. 273-294.

[7] G. Anastasi, M. Conti and W. Lapenna, "A Power Saving Network Architecture for Accessing the Internet from Mobile Computers: Design, Implementation and Measurements", The Computer Journal, Vol. 46, No. 1, pp. 3-15, Jan. 2003.

[8] M. Arlitt and T. Jin, "Workload Characterization of the 1998 World Cup Web Site", HP Laboratories Palo Alto, HPL-1999-35(R.1), September 1999.

[9] V. Baiamonte and C.-F. Chiasserini, "An Energy-efficient MAC layer Scheme for 802.11-based WLANs", Proc. of the IEEE Workshop on Energy-Efficient Wireless Communications and Networks (*EWCN 2004*), Phoenix (AZ), April 14-17, 2004.

[10] S. Bhattacharyya, C. Diot, R. Gass, E. Kress, S. Moon, A. Nucci, D. Papagiannaki, and T. Ye, "Packet Trace Analysis", Sprint Labs, Measurements from 2000 up to 2004, available on-line at http://ipmon.sprintlabs.com/packstat/packetoverview.php.

[11] A. Bakre and B.R. Badrinath, "Implementation and Performance Evaluation of Indirect TCP", IEEE Transactions on Computers, Vol. 46, No. 3, pp. 260-278, Mar. 1997.

[12] H. Balakrishnan, V.N. Padmanabhan, S. Seshan and R.H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", IEEE/ACM Transactions on Networking, Vol. 5, N. 6, Dec. 1997.

[13] P. Barford and M. Crovella, “Generating Representative Web Workloads for Network and Server Performance Evaluation”, Proc. of the 1998 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, pp. 151-160, June 1998.
[14] R. Bruno, M. Conti and E. Gregori, “Throughput Evaluation and Enhancement of TCP Clients in Wi-Fi Hot Spots”, Proc. of the First IFIP Working Conference on Wireless On-demand Network Systems (WONS 2004), LNCS 2928, pp. 73-86, Jan. 2004.
[15] R. Bruno, M. Conti and E. Gregori, “A simple protocol for the dynamic tuning of the backoff mechanism in IEEE 802.11 networks”, Computer Networks (37), pp. 33-44, 2001.
[16] F. Calì, M. Conti and E. Gregori, “IEEE 802.11 wireless LAN: capacity analysis and protocol enhancement”, IEEE Transactions on Networking, Vol. 8, No. 6, pp. 785-799, Dec. 2000.
[17] E.-Y. Chung, L. Benini, A. Bogliolo, Y.-H. Lu, and G. De Micheli, “Dynamic Power Management for Nonstationary Service Requests”, IEEE Transactions on Computers, Vol. 51, No. 11, pp.1345-1361, Nov. 2002.
[18] Cisco Aironet Wireless LAN Adapters 340 and 350 series, technical specifications, http://www.cisco.com/univercd/cc/td/doc/product/wireless/airo_350/350cards/pc350hig/pc_appa.htm#39209, 2005.
[19] M. Conti, G. Maselli, G. Turi and S. Giordano, “Cross-Layering in Mobile Ad Hoc Network Design”, IEEE Computer, Feb. 2004.
[20] M. Crovella e A. Bestavros, “Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes”, IEEE/ACM Transaction on Networking, Vol.5, No.6, pp. 835-846, December 1997.
[21] F. Donelson Smith, F. Hernandez Campos, K. Jeffay and D. Ott, “What TCP/IP Protocol Headers Can Tell Ue About the Web”, ACM SIGMETRICS 2001, pp. 245-256.
[22] J.P. Ebert, B. Stremmel, E. Wiederhold and A. Wolisz, “An energy-efficient power control approach for WLANs”, Journal of Communications and Networks (JCN), Vol. 2, No. 3, pp. 197-206, 2000.
[23] L.M. Feeney and M. Nilsson, “Investigating the energy consumption of a wireless network interface in an ad hoc networking environment”, Proc. of the 20$^{\text{th}}$ Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001), Anchorage (Alaska) Apr. 22-26, 2001.
[24] J. Flinn, S. Park and M. Satyanarayanan, “Balancing performance, energy, and quality in pervasive computing”, Proc. of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS02), pp. 217-226, Vienna (Austria), July 2-5, 2002.
[25] J.J. Garrett, “Ajax: A New Approach to Web Applications”, Adaptive Path Essay Archives, online available at http://www.adaptivepath.com/publications/essays/archives/000385.php, Feb. 2005.
[26] D.P. Helmbold, D.E. Long and B. Sherrod, “A Dynamic Disk Spin-down Technique for Mobile Computing”, Proc. of the 2$^{\text{nd}}$ Annual ACM International Conference on Mobile Computing and Networking (MobiCom ’96), pp. 130-142, Nov. 1996.
[27] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, “Hypertext Transfer Protocol – HTTP/1.1”, RFC 2616, June 1999.
[28] IEEE 802.11 WG, Draft Supplement to Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS), IEEE 802.11e/Draft 5.0, July 2003.
[29] IEEE standard for Wireless LAN- Medium Access Control and Physical Layer Specification, 802.11, November 1997.
[30] R. Krashinsky and H. Balakrishnan, “Minimizing Energy for Wireless Web Access with Bounded Slowdown”, Wireless Networks, Vol. 11, No. 1-2, pp. 135-148, Jan. 2005.
[31] R. Kravets e P.Krishnan, “Power Management Techniques for Mobile Communication”, Proceedings of the Fourth Annual ACME/IEEE International Conference on Mobile Computing and Networking (MOBICOM’98), 1998.
[32] J. Lee, C. Rosenberg, and E.K.P. Chong, “Energy Efficient Schedulers in Wireless Networks: Design and Optimization” to appear in the ACM/Springer MONET special issue on “Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks”.
[33] Y.-H. Lu, L. Benini and G. De Micheli, “Power-Aware Operating Systems for Interactive Systems”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 10, No. 2, pp. 119-134, April 2002.
[34] S. Nath, Z. Anderson and S. Seshan, “Choosing Beacon Periods to Improve Response Times for Wireless HTTP Clients”, Proc. of ACM International Workshop on Mobility Management and Wireless Access (*MobiWac’04*), Philadelphia (PA), Oct. 1, 2004.
[35] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, “Modeling TCP Throughput: A Simple Model and its Empirical Validation”, in ACM Sigcomm, 1998.
[36] A. Passarella, “Power Management Policies for Mobile Computing”, Ph.D. Thesis, Information Engineering Department, University of Pisa, Italy, Feb. 2005.
[37] S.H. Phatak, V. Esakki, B.R. Badrinath and L. Iftode, “Web&: An Architecture for Non-Interactive Web”, Proc. of the 2$^{\text{nd}}$ IEEE Workshop on Internet Applications (WIAPP01), S. Jose (CA), July 23-24, 2001.
[38] D. Qiao and K.G. Shin, “Smart Power-Saving Mode for IEEE 802.11 Wireless LANs”, Proc. of the 24$^{\text{th}}$ Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005), Miami (FL), March 13-17, 2005.
[39] N. Ramos, D. Panigrahi and S. Dey, “Energy-Efficient Link Adaptations in IEEE 802.11b Wireless LAN”, Proc. of the IASTED International Conference on Wireless and Optical Communications, *WOC 2003*, Banff, Alberta, Canada, July 14-16, 2003.
[40] T. Starnar, “Powerful Change Part I: Batteries and Possible Alternatives for the Mobile Market”, IEEE Pervasive Computing, Vol. 2, No. 4, pp. 86-88, Oct.-Dec. 2003.
[41] C. Schurgers, V. Raghunathan, and M. Srivastava, “Power management of energy-aware communication systems”, ACM Trans. Embedded Comp. Systems, vol. 2, iss. 3, pp. 431-447, August 2003.
[42] T. Simunic, L. Benini, P. Glynn, and G. De Micheli, “Event-Driven Power Management”, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol. 20, No. 7, pp. 840-857, July 2001.
[43] T. Simunic, L. Benini, P. Glynn, and G. De Micheli, “Dynamic Power Management for Portable Systems”, Proc. of the 6th annual international conference on Mobile computing and networking (Mobicom 2000), pp. 11-19, Boston, MA, 2000.
[44] M. Stemm and R.H. Katz, “Measuring and reducing energy consumption of network interfaces in handheld devices”, IEICE Trans. Fund. Electron, Commun. Comp. Sci. (Special Issue on Mobile Computing), Vol. 80, No. 8, pp. 1125-1131, 1997.
[45] W.R. Stevens, “TCP/IP Illustrated, Volume 1: The Protocols”, Addison-Wesley, 1994.
[46] H. Yan, R. Krishnan, S.A. Watterson and D.K. Lowenthal, “Client-Centered Energy Savings for Concurrent HTTP Connections”, Proc. of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (*NOSSDAV’04*), Kinsale, County Cork, Ireland, June 16-18, 2004.