Department of Information Engineering, University of Pisa, Italy

Ph.D. Thesis - XVII Cycle

Power Management Policies for Mobile Computing

Ph.D. Candidate

Andrea Passarella

Advisors

Prof. Giuseppe Anastasi

Prof. Luciano Lenzini

February 2005

Erica

Contents

Acknowledgements

I take this opportunity to express my gratitude to my supervisors, Prof. Giuseppe Anastasi and Prof. Luciano Lenzini of the Information Engineering Department (University of Pisa), and to Dr. Marco Conti and Dr. Enrico Gregori of the Institute for Informatics and Telematics (CNR) of Pisa. Their support has been invaluable, and I owe them my professional growth during these years. Also, I would like to thanks them for creating a so friendly working environment, where people can actually express themselves at best.

I would like to thanks also all the guys working together with me during these years, especially Emilio, Franca, Gaia, Giovanni, Leli, Luciana and Raffaele.

Last but not least, I would like to thanks my family. This achievement belongs to them too.

Part I.

Background

1. Introduction

1.1. Problem Statement

The proliferation of mobile computing and communication devices is producing a revolutionary change in our information society. Laptops, smart-phones and PDAs, equipped with wireless technologies, support users in accomplishing their tasks, accessing information, or communicating with other users anytime, anywhere. Projections show that in few years the number of mobile connections and the number of shipments of mobile terminals will grow yet by another 20-50 percent [55]. With this trend, we can expect the total number of mobile Internet users soon to exceed that of the fixed-line Internet users.

Currently, most of the connections among mobile devices and the Internet occur over fixed infrastructure-based networks, which are extended with a wireless last hop. For example, cellular networks (GSM, GPRS or UMTS) provide a wireless link between mobile phones and a base station, while laptops connect to the Internet via Wi-Fi Access Points (i.e., inside Wi-Fi hotspots). In particular, installations of Wi-Fi hotspots are nowadays more and more frequent, for example in company and education buildings, coffee shops, airports, and so on. Figure 1.1 gives a pictorial representation of this scenario. Specifically, a network operator covers a limited-size area (i.e. a "hotspot") with Access Points connected to the legacy fixed Internet. Users equipped with mobile devices (such as laptops or PDAs) can connect to Internet services (e.g., visit a Web site) wirelessly. Seamless wireless connectivity in such environments is of great value for Internet users. Moreover, the low cost of devices (i.e., Access Points and wireless cards), and the Wi-Fi hotspot ease of installation and setup is boosting their diffusion. Wi-Fi is really a Plug&Play technology, that can greatly improve the Internet-user experience.

Despite their increasing popularity, many technical problems have to be fixed for Wi-Fi hotspot to guarantee sufficient Quality of Service to Internet users. Among them, one of the most critical is

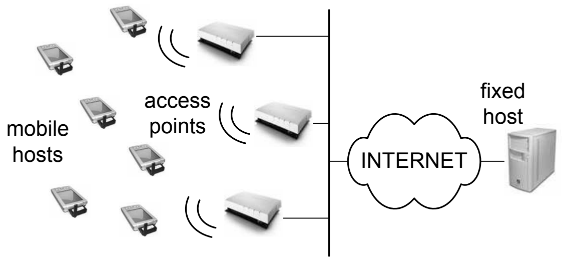


Figure 1.1.: The reference environment: a Wi-Fi hotspot.

power management at mobile devices. To allow users mobility, devices must be battery-supplied. It is common experience that current mobile devices (laptops, PDAs, etc.) can operate just for few hours before the battery gets exhausted. Even worse, the difference between power requirements of electronic components and battery capacities is expected to increase in the near future [51]. In a nutshell, power management for mobile devices is a must for the development of mobile computing scenarios, and each (hardware or software) component of a mobile device should be designed to be energy efficient. The networking subsystem is one of the critical components from the power management standpoint, as it accounts for a significant fraction of the total power consumption (around 10% for laptops, and up to 50% for small hand-held devices, such as PDAs [7, 39]).

This work aims at exploring possible solutions for this key issue. We focus on best-effort applications, i.e. applications without real-time requirements. Example of such applications are Web, e-mail and file transfer that represent the lion's share of the today-Internet traffic [15]. These applications generate a bursty traffic, as shown in Figure 1.2. Specifically, data are exchanged between a mobile and a fixed host as a sequence of bursts spaced by idle times (referred to as User Think Times). Within bursts, packets are separated by idle times (referred to as interarrival times). While User Think Times are generated by the human behavior (e.g., in the Web case a burst represents a Web page, and a User Think Time the time spent by the user reading the page), interarrival times stems from computer interactions due to network protocols, and hence they are much shorter. It is well-known that the main source of energy wastage in such a traffic is the fact that the wirless interface of the mobile host remains powered on during idle times. Therefore, the main power-saving technique consists in turning it into a low-power mode (possibly, switching it off) during these time intervals. Exploiting this technique is not trivial. Indeed, the length in time of idle times is usually unknown in advance. Furthermore, re-activating the wireless interface from any low-power mode has an energetic cost. For short idle times, letting the wireless interface on instead of turning it in a low-power mode could be even more energy efficient. Finally, if several low-poewr modes are available, the most appropriate one should be chosed each time the wireless interface becomes idle.

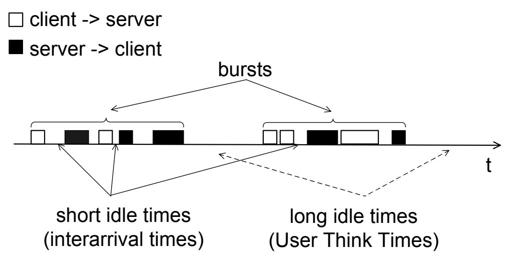


Figure 1.2.: The reference network traffic.

1.2. Our approach to Power Management

In this work we design several power-saving solutions implemented in network protocols. The core of these solutions is detecting idle times, estimating their length, and managing the wireless interface accordingly.

We firstly propose two middleware approaches, named PS-Web and PS-WiFi, respectively. These approaches are based on detecting idle times and estimating their length in time. The wireless interface is switched off for the (predicted) duration of the idle time. The difference between PS-Web and PS-WiFi is that PS-Web is application-*dependent*, i.e., it exploits a-priori knowledge about the application behavior to detect and estimate the lenght of idle times. On the other hand, PS-WiFi is application-*independent*, and dynamically intercepts the applications behavior by monitoring their traffic. We design these systems, and provide the related network protocols. Then, we evaluate them extensively. Specifically, we use real-Internet prototypes and analytical models to achieve a clear understanding of their behavior.

Then, we compare these middleware solutions and the de-facto standard MAC-level policy for Wi-Fi hotspots, i.e. the Power-Saving Mode (PSM) of the IEEE 802.11 standard. The advantage of middleware-layer policies is that they can exploit a clar knowledge about the application behavior, which is not available at the MAC level. Furthermore, since they do not rely on any particular wireless technology, PS-Web and PS-WiFi are highly portable. On the other hand, 802.11 PSM can exploit specific low-power modes of the wirless interface that are defined by the standard. On the other hand, for the sake of portability, PS-Web and PS-WiFi can just switch the wireless interface off.

To compare the MAC- and the middleware-layer approaches, we extensively evaluate the power-saving performance of PSM. Overall, the power-saving achieved by the two approaches is comparable. A very interesting outcome is that these approaches are somewhat complementary, since PSM performs better during bursts (i.e., during interarrival times), while PS-Web and PS-WiFi perform better during User Think Times. Therefore, we define a Cross-layer Power Manager (CPM) that integrates the 802.11 and middleware-level mechanisms included in PS-Web and PS-WiFi. We evaluate CPM in depth, and show that it significantly improves the performances of “single-layer” policies in terms of power saving. This results highlight how a cross-layer design is powerful in this environment. Furthermore, CPM does not require hardware modifications, and can be entirely implemented at the mobile host. Hence, it represents a very interesting power-saving solution that is suited to be implemented in current 802.11 hotspots.

2. Thesis Contribution

The main contribution of this work is to provide an effective framework for addressing the power-saving problem in wireless hotspots. We focus on reducing the power consumption of the networking subsystem, since this is the most energy-hungry part of current mobile-computing devices (e.g., PDAs). We design power-saving solutions that are transparent both to the application level, and to the wireless technology adopted in the hotspot. As such, they do not require modifications either in the application code, or in the wirless hardware. Clearly, this is a key advantage to quickly deploy real systems based on our solutions.

In this work we provide a comprehensive view of the energy-conservation problem in wirless hotspots. We show that a single policy does not fit any possible application scenario. Hence, we study different policies, that are best suited for each scenario taken into consideration. We deeply evaluate each solution, providing evidence of their strengths and weaknesses.

We envisage two orthogonal classifications. On one hand our power-saving policies can be classified as application-dependent and application-independent policies. Application-dependent policies, exploit a-priori information about the application behavior, and are the best option when the set of application running in the wireless environment can be known at design time. We propose application-independent policies as well that are just slightly less effective than application-dependent ones, and are suitable to support any kind of best-effort application (e.g. Web, e-mail, file transfer, . . .). Therefore such policies can be used in general-purpose environments.

On the other hand, our policies can be classified as pure-middleware and cross-layer policies. Pure-middleware policies are not aware of the particular wireless technology they will support, and hence they do not exploit any technology-dependent power-saving feature. As such, these policies are highly portable. Furthermore, we show that more effective policies can be designed when the wireless technology operating in the wireless hotspot is known in advance. Specifically, we focus on the leader wireless technology, i.e., IEEE 802.11, and we leverage a cross-layer approach to power management. We show that cross-layer policies, operating both at the MAC- and middleware-layer, significantly outperform policies that operate just at a single level in the protocol stack. To achieve a complete comparison, in this work we develop a detailed analytical and simulation model of the 802.11 Power-Saving Mode, and we provide a deep characterization of the PSM behavior. This is another major contribution of our work, since, to the best of our knowledge, this is the first effort in the literature to provide a such a detailed characterization of 802.11 PSM.

To summarize, in this work we address the power-saving problem for wireless hotspots in several common environments. For each environment, we design and investigate effective policies that significantly reduce the power consumption of mobile devices. Therefore, this work provides comprehensive guidelines for deploying power-saving systems in real wireless environments.

3. Thesis Layout

The remainder of the thesis is organized as follows. First of all, we survey works related to ours (Section 4). Then, the pure middleware-layer solutions are presented and extesively evaluated in part II. Specifically, in Chapter 5 we design and evaluate an application-dependent solution (PS-Web) tailored to Web applications. Firstly, we provide the PS-Web design (Section 5.3) in the reference environment (Section 5.2). PS-Web is compared against different power-saving strategies (Section 5.4), and then it is extensively evaluated (Sections 5.5 $\div$ 5.7). Chapter 6 is devoted to present and evaluate an application-independent solution (PS-WiFi). PS-WiFi is presented in Section 6.2, and evaluated through experiments run in several application environments (Section 6.3). Then, an analytical model of PS-WiFi is derived (Section 6.4), and used to better understand the PSWi-Fi behavior (Sections 6.5 and 6.6).

Part III is devoted to presenting and evaluating our Cross-Layer approach to power-management. Specifically, Section 7.3 provides an overview of the 802.11 Power-Saving Mode (PSM). A PSM analytical model is presented in Section 7.4. This model, along with simulation results, is used to extensively evaluate the PSM performance in terms of power-saving (Section 7.5). Then, our Cross-Layer Power Manager is presented and evaluated in Section 7.6.

Finally Chapter 8 draws our conclusions about Power Management for Mobile Computing, and provides some future research directions.

4. Related works

Uderstanding and enhancing the performance of wireless LANs, mainly in terms of power saving, has deserved increased attention in the last few years. Some works highlight limitations of 802.11 and propose enhancements. Other works propose power-saving policies that are not specifically tailored to 802.11, but that can be applied also to this technology. For ease of reading, in the following of this section we follow this broad classification to present these works.

4.1. 802.11 characterizations

Krashinsky and Balakrishnan [38] carry out a simulation analysis of 802.11 PSM in presence of Web browsing traffic. In particular, they consider a single mobile user (i.e., no congestion) inside the hotspot. They show that PSM can save around 90% of the energy spent by the wireless card at the cost of increased delay in the Web-page downloads. To cope with this problem, the authors 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. It should be pointed out that the problem of increased Web delays arises for very short Round Trip Times between the mobile and the fixed host, (i.e., below 80 ms), and is far less marked for increasing Round Trip Times. In this work, we focus on a broader range of Round Trip Times, for which such additional delays are more tolerable. Furthemore, with respect to CPM, BSD just uses the sleep state of the mobile host to save energy, while PS-Web, PS-WiFi and CPM switch the mobile host off during User Think Times. As discussed in the third part of this work, this choice allows greater energy saving. On the other hand, during burst-download phases, BSD improves PSM since it reduces the additional delay of Web-page donwloads. From a power-saving standpoint, during burst download phases BSD performs similarly to PSM, and hence it outperforms both PS-Web and PS-WiFi. However, it should be noted that BSD requires non-trivial hardware modifications, and operates only in 802.11-based networks. As far as CPM, it should be noted that, thanks to its flexible design, CPM is able to exploit any power-saving policy during burst-download phases. Therefore, BSD could be used in the Cross-layer Power Manager in that cases where it outperforms PSM. Finally, in this work we analyze PSM with respect to several parameters (such as a broader range of RTT values, the average bursts size, the packet loss probability and the WLAN congestion), which are not taken into consideration in [38].

Similar remarks apply also to [44]. Specifically, [44] proposes the *Dynamic Beacon Period* algorithm (DBP), aimed 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 Beacon frames for each mobile host. Several scalability issues are not addressed in [44] that are key points to fairly evaluate DBP. However, since DBP operates during burst-download phases, it can be integrated into CPM.

Anand et al., [2] carry out an experimental evaluation of 802.11 PSM on HP IPAQs. They primarily focus on the traffic generated by applications using network file systems such as NFS and Coda. Their results confirm [38], with respect to the additional delay added by the 802.11 PSM. A very interesting outcome is that, for high-end devices such as laptops, this may lead to an *increase* in the total energy consumption of the mobile device. The additional delays make the other laptop components being active for longer time, thus overwhelming the energy saved in the networking subsystem. To overcome this problem, they propose Self-Tuning Power Management (STPM). 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 knowledge about the power consumption of the whole device, to manage the wireless interface appropriately. In [2] the choice is just between activating the PSM or not, but the STPM system is flexible enough to implement more sophisticated policies. The energy saving achieved by STPM is 21% with respect to the standard PSM, and the additional delay is reduced by 80%. Our work shares some similarity with [2]. As STPM does, our Cross-layer Power Manager sits on top of different power management policies, and dynamically chooses the most appropriate one. Furthermore, both STPM and our Cross-layer Power Manager can be implemented in the Operating System of the mobile host, and can be deployed in current Wi-Fi hotspots. The main difference between [2] and our work is that all our policies are simpler, and do not require explicit collaboration from the applications, i.e., no modifications to the application code is required. We believe that this feature is very important, at least in a medium-term perspective. Other differences between our work and [2] are that [2] assumes very short RTTs between the mobile and the fixed host, and focuses on a specialized scenario. Specifically, authors focus on distributed interactive software deployment supported by network filesystems. Network filesystems are implemented by means of RPCs, that involve many short request/response interactions among the clients and the server. The impact of PSM additional delay on such traffic pattern is severe. In our work we focus on Web-like traffic patterns, which are more representative of the typical hotspot use. Web requires less interactions between the clients and the server, and the effect of additional delays is not heavy.

Finally, [14, 49] propose power-saving policies for 802.11 WLAN that are orthogonal to the work presented in this paper. Specifically, [14] puts the mobile host in the sleep mode during MAC-level contention periods. On the other hand, [49] reduces the energy spent during transmissions by dynamically adapting the fragmentation threshold, the transmission power, and the retry limit. Both of these works are extensions to the standard PSM that work during burst-download phases, and hence they can be easily included into our Cross-layer Power Manager.

4.2. Power-saving policies for generic wirless LANs

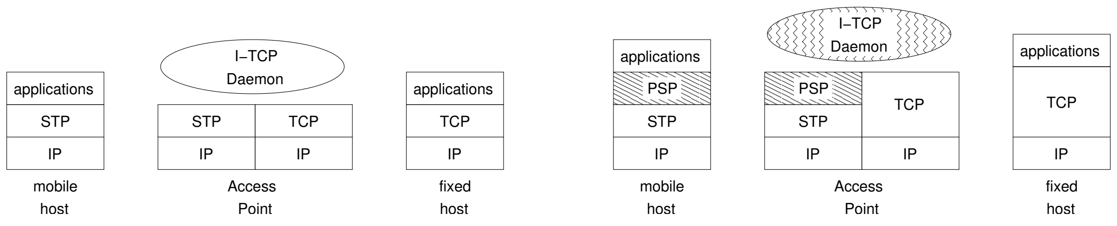
Other works face the power-management problem in WLAN environments, but do not focus on a specific wireless technology. [39, 52, 54, 41] propose power managers implemented in network protocols. Specifically, [39, 52, 54] use inactivity timeouts to decide when to switch off the wireless interface. Timeout values are fixed, and depend on the particular application. [39] relies on an Indirect-TCP architecture in order to buffer at the Access Point possible packets arriving while the mobile host is disconnected. Instead, [52] avoids any support from the Access Point, and exploits knowledge about the application behavior to avoid missing packets. Also [54] uses a pure client-centric approach, i.e., no support from the Access Point is exploited. Specifically, an approach similar to our is used, in the sense that interarrival-time lengths are estimated on-line. Furthermore, inactivity timeouts are used to detect User Think Times. With respect to our work, no support from the Access Point is exploited, and hence packets that may arrive while the mobile host is disconnected are lost. Inactivity timeouts are also used by our Cross-layer Power Manager. However, in our system they are dynamically adjusted based on the status of the network path. [41] assumes a PSM like algorithm, in the sense that the Access Point periodically signals frames buffered for mobile hosts. However, in their system the Access Point signals only the frames that will be transmitted in the next period. Hence, other stations may sleep, whether they have packets buffered at the Access Point or not. Authors analyze several scheduling algorithm to choose which frames to announce, so as to minimize the overall energy consumption. Modifying the way the Access Point manages the download traffic has also been proposed in [21] as a mean to reduce congestion in a WLAN, and optimize the hotspot performance in terms of throughput. Actually, this is a very promising way also from a power-saving point of view. It should be noted that such policies can be implemented in our Cross-layer Power Manager, since they are focused on the burst download phases.

[42, 23, 50] advocate power management at the Operating System level. [42] exploits on-line application-level hints to decide when to shut the network interface down. Hence, this system requires modification of the applications' code. The authors of [23, 50] formulate the power-management problem as a linear program, where the objective is minimizing the power consumption of a particular component, and the maximum tolerable performance degradation (for example in terms of additional delays) is the constraint. Then, they derive optimal power 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 our power-saving systems.

Finally, other approaches to power management include transmission power control techniques [28], or complete novel application-level architectures [47, 37, 30, 31]. Specifically, [47] propose "Web&", a multi-tier architecture for disconnected Web transactions. User of a mobile host describes the service she requires to an agent running at the Access Point. For example, she declares the destination, date and time needed for a flight. The agent becomes responsible of exploring the Web and finding flights that could fit for her requirments. While this process is ongoing, the mobile host can disconnect. Eventually, once the mobile host connects again, the agent at the Access Point provide the search results to the user, who can choose the most appropriate solution. Though interesting, this solution is customize to non interactive applications, and requires significant modifications to the application architecture. [37] propose transcoding as a mean for reducing power consumption in mobile-Web applications. Specifically, Web page components (e.g., images) are encoded at lower quality when downloaded by mobile hosts. That way the size of Web pages is reduced, and this results in energy conservation. However, it should be noted that the main source of energy consumption is during idle times, while the burden related to transferring data is usually not that much. Nevertheless, such solutions can be seen as complementary to systems aimed at reducing the effect of idle times. A trade off between power consumption and QoS is the goal of [30, 31]. The system they propose leverage collaboration among applications and the operating system to save energy. User of a mobile host declares the expected amount of time the battery should last. Based on this information, the system, for each operation, suggests to applications the quality that should be used in order to meet that goal. For example, a video stream could be dowloaded and played at a lower quality if the battery is required to last for several hours. Such is very interesting, and could be included in future generations of applications and operating systems. The solution we provide here is somewhat "quick and dirty": it can be deployed in current Wi-Fi installations without requiring significant modifications, and is still able to conserve most of the energy currently spent in networking activities.

Part II.

A Pure Middleware-Layer Approach to Power Management



(a) Indirect-TCP model (b) Power-Saving model

Figure 4.1.: The Indirect-TCP and the Power-Saving model (evidence on added and modified components)

Within the framework described in the previous part, we have envisaged two possible approaches, operating at the middleware layer: the *application-dependent* approach and the *application-independent* approach.

Both approaches share some basic architectural design aspects. They both rely upon a network architecture based on the Indirect-TCP model 4.1(a). The mobile computer connects to a fixed host (e.g., a Web server) through a third entity (the Access Point) located at the border between the wireless and wired networks. With respect to the traditional TCP/IP architecture, the transport connection between the mobile host and the fixed host (e.g., a Web server) is split into two parts. The first one connects the mobile host with the Access Point, while the second connects the Access Point and the fixed host. At the Access Point a software agent (the Indirect-TCP Daemon) relays data between the two connections. The original Indirect-TCP model [16] uses the TCP protocol also on the wireless part of the connection. On the contrary, we use the *Simplified Transport Protocol* (STP), which provides a connectionless, reliable type of service. It has been shown [17] that such a protocol is much more suited to the single-hop wireless environment.

Power-saving functionalities are included in such an architecture by defining power-saving protocols between the Access Point and the mobile host, as shown in Figure 4.1(b). As highlighted above, our solutions operate at the middleware layer, and hence the protocols we design work on top of the transport protocol (see the following sections for more details). The design of our power-saving solutions is based on the following remarks. To save energy the mobile host periodically switches the network interface off. While disconnected, data coming from the Internet and destined to the mobile host are temporarily stored by the Indirect-TCP Daemon. To decide when and how long the network interface should be off, both the application-independent and the application-dependent approaches dynamically estimate the traffic behavior (packet inter-arrival times, idle periods, etc.). As highlighted in the following of the work, in some cases the original Indirect-TCP Daemon is slightly modified to support the power-saving functionalities. Switching off the network interface actually reduces the energy consumption but can heavily increase the user response time (e.g., in the case of the Web application, the elapsed time between the generation of a request from the browser for the retrieval of a Web page, and the rendering of that page at that browser's site), thus negatively affecting the QoS perceived by the user. Thus, a trade off between these two orthogonal performance figures must be reached.

The basic difference among the two approaches resides in the algorithms used to decide when, and how long, to keep the network interface off. In the application dependent approach the semantic of the application(s) is exploited. For example, in the case of the Web application, the power-saving policy is aware of the traffic pattern related to the download of a Web-page. Based on this knowledge, and by spoofing the application-level traffic, this policy predicts the near-future network activity of the mobile host, and manages the wirless interface accordingly. On the other hand, the application independent approach is based on algorithms that do not rely upon any a priori application semantic but try to dynamically intercept the behavior of the active application(s). By "learning" on-line how the application(s) behave, this policy predicts the future traffic pattern, and manages the wireless interface accordingly.

The application dependent approach is tuned to the specific application, and hence it performs the best from a power-saving point of view. However, it requires a different power-saving module for each application, and – in the case of concurrent applications – a further module to coordinate the application-specific ones. The application-independent approach is much more flexible. Specifically, it supports any type of (non real-time) application, and works also in the case of concurrent applications. It is therefore interesting to compare its performance with those of the application dependent approach that constitute a target reference.

As a rule of thumb, it can be said that application-dependent approaches are best suited to run in dedicated environments, where the set of applications that will run in the system can be defined at design time. In such a scenario, using the application-dependent approach guarantees the best peformance both in terms of power saving, and in terms of QoS. On the other hand, in open environments, where users may run arbitrary applications, the application-independent approach is the best candidate. It still achieves very good performances both in terms of power saving and in terms of QoS, but supports any application without interventions by the network operator.

In the following, we deeply analyze the performance of both approaches. As far as the application-dependent approach, we analyze a system (PS-Web), designed to support Web applications. As far as the application-independent approach, we analyze a system (PS-WiFi) that supports any type of best-effort traffic. We test this systems with respect to Web applications, to e-mail applications, and also in the case of concurrent applications, i.e., Web and e-mail.

5. Application-Dependent Power Management

5.1. Overview

In this section we define a new architecture, throughout referred to as *PS-Web* (Power-Saving Web), which allows mobile users to exploit Internet Web services with a QoS similar to the one provided by the legacy network architecture based on the TCP/IP protocol stack, but with a significant reduction in the energy consumption. The PS-Web architecture is based on the Indirect-TCP model [16], i.e., the TCP connection between the browser and the Web server is split into two connections: one between the browser (on the mobile computer) and an Access Point (at the border between the wireless and wired networks), and the other one between the Access Point and the Web server. Unlike the solution proposed in [39], however, a simplified transport protocol is used between the mobile host and the Access Point. Furthermore, inactivity timeouts and sleeping times used to switch off and on the network interface are not fixed – as in [52] and [39] – but are adjusted dynamically based both on information about the past history collected on-line and on statistical models of Web traffic pattern available in the literature. The Access Point works as a Power Saving Proxy Web, i.e., a Proxy Web with power saving support for mobile users. Specifically, it implements a pre-fetching mechanism.

In order to evaluate the PS-Web effectiveness, we compare the performance of four different power-saving strategies aimed at reducing the energy consumed during a Web-page download. The first strategy is a pure Indirect-TCP (I-TCP) architecture. With respect to the legacy TCP architecture, this solution improves the throughput achieved by the mobile host, thus reducing the transfer-time. Hence, it indirectly contributes to power saving even though no energy-management mechanism is explicitly introduced in the system. Explicit energy management is included in the other policies we consider, all obtained by enhancing the I-TCP architecture. The *local* strategy switches the wireless interface off when the user is reading the Web page, i.e., it exploits information that are locally available at the client browser. On the other hand, PS-Web, referred to as the *global* strategy, in addition to local information, exploits statistical information about Web traffic. Finally, an ideal (unfeasible) strategy that guarantees the minimum power consumption is also considered. Throughout this part of the work, the ideal and I-TCP strategies provide the lower and upper bound for energy consumption, respectively.

We implemented the feasible power-saving strategies and tested them extensively in a real Internet scenario. Our performance study is based on two main performance figures: I_{ps} and I_{pd}. I_{ps} is used as a power saving index. It measures the energy consumption of a specific strategy expressed as a percentage of the energy consumption related to I-TCP strategy. I_{pd} measures the impact of the power saving strategy on the User Response Time (URT), i.e., the time interval elapsed from a

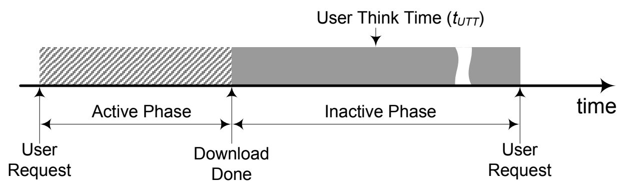


Figure 5.1.: The Web-page download as an Active and an Inactive Phase.

user request for a Web page to its rendering on the mobile device.

The experimental results show that the global strategy exhibits the best achievable performance. It saves, on average, 88% of the energy consumed by the I-TCP approach and has a negligible impact on the URT (the URT increase is of 0.2 sec on average, and is below 1.8 sec with probability 0.9).

5.2. System Model

The power-saving strategies evaluated in our system are application-dependent, i.e., they exploit the application semantic to optimize the energy consumption. Hence, as a preliminary step, it is necessary to characterize the traffic profile generated by Web browsing.

Many papers in literature provide mathematical Web traffic characterizations [26, 13, 12, 25, 19, 18], and show that, with an appropriate analysis of the Web servers logs, it is possible to model the Web user behavior [25, 19].

5.2.1. Single user's traffic model

The activity of an individual user can be represented as a series of successive requests for Web pages. As shown in 5.1, each request causes a two-phase process. During the first phase, the Web page is downloaded from the server to the client while, in the second phase, the user reads the contents. The first phase is typically named Active Phase because during this time interval data flow on the network. The second phase is referred to as Inactive Phase because there is no network activity.

The Inactive Phase is composed by a unique time interval (t_{UTT} in Figure 5.1). This time interval is known as the *Inactive Off Time* or *User Think Time* (UTT), and is typically longer than 30 seconds (i.e., it is practically much longer than the Active Phase length). User Think Times are distributed according to a Pareto law [25, 19]:

$$p\left(t_{UTT}\right) = \alpha k^{\alpha} t_{UTT}^{-(\alpha-1)},\ t_{UTT} \geq k,\ \alpha = 1.5,\ k = 30\ , \tag{5.1}$$

where k is the *scale parameter* and α is the *shape parameter*.

Figure 5.2 provides a graphical representation of a typical Active Phase. A Web page usually consists of a set of files: an HTML *main file* and a number of *embedded files*. Specifically, the main

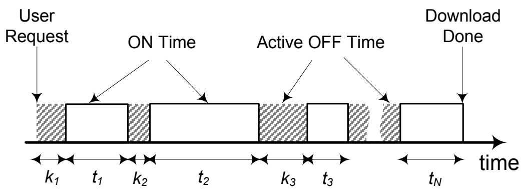


Figure 5.2.: The Active Phase as a sequence of ON and OFF Times.

file contains the page textual information, the names of the embedded files and a description of the page layout. The browser transfers the whole set of files and arranges them in the page.

The Active Phase can be seen as a sequence of N ON Times (t_i in Figure 5.2) and N Active OFF Times (k_i in Figure 5.2), where N is a random variable. The main file is transferred during the first ON Time. Then, the transfer of each embedded file occurs in subsequent ON times. ON times are usually separated by OFF times. Among the others, an OFF Time includes the time required by the client to prepare HTTP request(s). These OFF Times are typically referred to as *Active OFF Times*, to distinguish them from the User Think Times.

The length of a single ON Time can be described as follows:

$$t_i = \frac{B_i}{\gamma_i} + \delta_i = \frac{D_i + h_i}{\gamma_i} + \delta_i \ , \tag{5.2}$$

where:

- ✧ B_i is the size (in bytes) of an overall HTTP transaction needed to fetch a file. Specifically, it includes the file size (D_i), and the headers of all packets containing the HTTP request(s) and HTTP response (h_i).
- ✧ γ_i is the throughput experienced during this transaction.
- ✧ δ_i depends on the specific HTTP version, and may include the sum of the network Round Trip Time (RTT), and the time needed by the Web server to process an HTTP request.

It must be pointed out that γ_i and δ_i depend on the network traffic conditions, while h_i can be closely approximated with a constant value. D_i depends on the distribution of the Web file sizes. In the literature, the file size is modeled according to a hybrid distribution: the tail and the body are modeled according to Pareto (see Equation 5.1) and lognormal distributions (see Equation 5.3), respectively.

$$p\left(x\right) = \frac{1}{\alpha x\sqrt{2\pi}} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}} \ . \tag{5.3}$$

The parameters of lognormal (i.e., μ and σ) and Pareto (i.e., α and k) distributions, as well as the cutoff value between the two distributions, depend on the set of files available at the Web server.

Active OFF Times (k_i in Figure 5.2), are typically modeled according to a Weibull distribution:

$$p\left(t\right) = \frac{bt^{b-1}}{a^b} e^{-\left(\frac{t}{a}\right)^b} \ . \tag{5.4}$$

The Weibull parameters do not depend on the particular Web site. Typical values are a=1.46, b=0.382 [19].

Finally, N denotes the number of Active OFF and ON Times (see Figure 5.2). Obviously, $N = 1+e$, where e is the number of embedded files, and 1 corresponds to the main file. The number of embedded files, e, is typically modeled according to a Pareto distribution, where α and k parameters depend on the specific Web server [19].

As a final remark, it has been shown in [25] that the Web-page download statistical models are strictly related to the self-similarity property of Web traffic. Since this is a *structural* property of the Web traffic, the characterization provided in this section do not depend either on the Web contents or on the user access patterns.

5.2.2. General energy consumption model

In this section we introduce a model for the energy consumption in a mobile Web access scenario. As explained in the Introduction, the energy consumption is approximately proportional to the time during which the wireless interface remains in the ON state. Therefore, hereafter we will measure the energy consumption as the wireless interface ON time. Equation 5.5 provides the energy, C, consumed for downloading a Web page:

$$C = \sum_{i=1}^{N} \left(\frac{D_i + \beta_i}{\gamma_i} + \tau_i \right) + A + U + m \cdot t_{so} \ . \tag{5.5}$$

where:

- ✧ β_i measures the overhead in bytes introduced in the i-th file download. In addition to the size of the HTTP request and response headers (in Equation 5.2), β_i also includes specific overheads associated with the implemented power-saving strategy (if any).
- ✧ γ_i is the throughput experienced in the file transfer (see Equation 5.2).
- ✧ τ_i is the overhead in time related to the download of the i-th file. Specifically, in addition to δ_i (see Equation 5.2), it also includes specific time overheads associated with the implemented power-saving strategy (if any).
- ✧ A is the contribution to the energy consumption due to the Active OFF Times. This contribution is the sum of the Active OFF Times ($A = \sum_{i=1}^{N} k_i$) if no power-saving strategy is implemented. Power-saving strategies typically reduce this quantity.
- ✧ U is the contribution to the energy consumption due to a User Think Time. This exactly corresponds to the User Think Time if no power-saving strategy is implemented. The aim of power-saving strategies is to reduce it.
- ✧ $m \cdot t_{so}$ is the total contribution to the energy consumption due to the transients caused by the off-on switching of the wireless interface. When the wireless interface is turned on, there is a transient period during which it consumes energy but it cannot be used for data transfer.

Symbol	*Explanation*
N	number of files in a Web page
k_i	length of the i-th Active OFF Time inside the Active Phase
t_i	length of the i-th ON Time inside the Active Phase
t_{UTT}	length of the User Think Time after the download of a Web page
B_i	dimension of the HTTP transaction needed to fetch the i-th file of a Web page
D_i	dimension of the i-th file of a Web page
h_i	dimension of the HTTP request and response headers used in the i-th HTTP transaction
γ_i	average throughput experienced by the mobile host during the i-th HTTP transaction
γ	maximum throughput available on the wireless link
δ_i	sum of the network Round Trip Time and the time needed by the Web server to process the i-th HTTP request.
β_i	overhead in bytes introduced in the i-th file download
τ_i	overhead in time introduced in the i-th file download
A	Active OFF Times contribution to the energy spent to download a Web page
U	User Think Time contribution to the energy spent to download a Web page
m	number of times the mobile host wireless interface switches from off to on during the Active Phase
t_{so}	time interval needed by the mobile host wireless interface to switch from off to on
g	number of residual-transfer-time estimates provided by the Access Point during the Active Phase of a Web-page download
s	number of residual-transfer-time estimates greater than t_{so} $(s \leq g)$

Table 5.1.: Symbols used in the model

In Equation 5.5 t_{so} denotes the length of the transient period (typically, and throughout this work, 100 ms), while m is the number of off-on transitions during the Web-page transfer[1].

✧ D_i and N define the traffic characteristics (see Table 5.1) and do not depend on the particular power-saving strategy.

For reader convenience, in Table 5.1 we summarize the symbols that are used hereafter.

5.3. PS-Web architecture and protocols

A typical mobile scenario is depicted in Figure 5.3. Specifically, with respect to the general case shown in the Introduction, we here consider just a single mobile host connecting to the Web through a Wi-Fi hotspot. Although very simple and costless, a legacy TCP-based solution is prone to various drawbacks that heavily impacts the energy consumption at the mobile host.

[1]The m value depends on the specific power-saving strategy. Obviously, when no power-saving strategy is implemented, m=0.

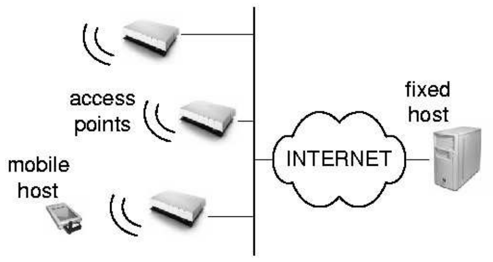


Figure 5.3.: A typical mobile environment (single-user case).

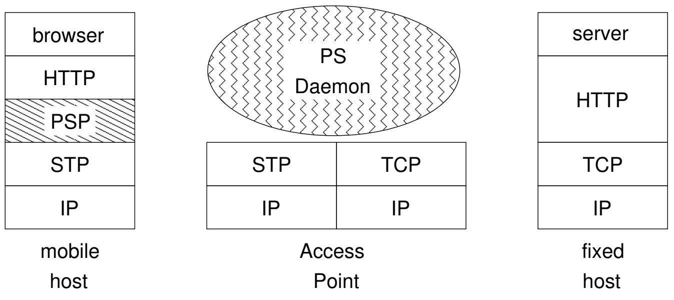


Figure 5.4.: PS-Web network architecture (evidence on added and modified components).

1. The TCP congestion control wrongly interprets losses in the wireless link as congestion signals. Hence, the overall throughput is usually low and the wireless network interface at the mobile host remains idle for most of the time.

2. Congestions in the wired networks limits the throughput in the wireless link as well. The overall effect is the same as in 1.

3. The ON/OFF behavior of Web traffic forces the wireless network interface to be inactive for long time intervals.

To overcome these problems we exploited a network architecture based on the Indirect-TCP model [16], with the Simplified Transport Protocol operating between the mobile host and the Access Point. The Indirect-TCP model eliminates problems related to point 1 above. However, bottlenecks in the Internet might still cause a low transfer rate in the wireless link. To overcome this second problem, we use pre-fetching of Web pages at the Access Point. Embedded files – if any – are requested to the remote server even without an explicit request from the user and will be transferred to the mobile host, on request from the mobile host itself. This approach allows to transfer embedded files on the wireless link at full speed, irrespective of the throughput available in the wired connection. At the mobile host side, pre-fetching is managed by the PSP (*Power Saving Protocol*) module. At the Access Point side, it is handled by the *PS-Daemon* (see Figure 5.4). This is the I-TCP Daemon enriched with pre-fetching and power management mechanisms.

Finally, with reference to point 3 above, it can be observed that, by grouping the transfer of the embedded files on the wireless link in a single burst, Active OFF Times can be compacted in an unique long OFF Time. This reduces significantly the time during which the network interface must be on. At the mobile host, the PSP layer is responsible for identifying the beginning of the Inactive OFF Times, and turning the network interface off until a new request from the browser arrives.

5.3.1. Power Saving Protocol (PSP)

The PS-Daemon can be seen as made up of two components. The upper level component interacts with the HTTP modules at the mobile host and fixed server, and implements the same functionalities of a Proxy Web. The lower level component implements power management by interacting with the PSP module at the mobile host via the Power Saving Protocol. Therefore, the PS-Daemon can be regarded as a Proxy-Web with power saving support.

Since we are interested in power management, in the following we shall focus on the PSP protocol. The pseudo-code 1 shows the actions performed at the mobile host (left-hand side) and the Access Point (right-hand side), respectively.

Upon receiving the main file from the remote server, the PS-Daemon forwards it to the mobile host, together with an estimate of the residual transfer time (see below), i.e., the time needed to fetch the embedded files from the server (lines 1-5). Upon receiving such an estimate the mobile host turns the network interface off for the corresponding time interval (lines 4-8). Possible requests for embedded files generate by the browser in the meanwhile will be blocked by the PSP layer until the network interface is turned on again (lines 9-14).

When the time interval has elapsed, the PSP module at the mobile host turns the network interface on and sends requests for embedded files, if any, to the PS-Daemon (lines 15-18). The PS-Daemon has already fetched these files from the server and can thus send them back to the browser (lines 6-8).

When the Web page is completely available at the mobile host the PSP module turns the network interface off (lines 19-20) until a new request arrives from the user.

The architecture depicted in Figure 5.4 is completely transparent to the application and the HTTP protocol, respectively. Like any other Web proxy, the PS-Daemon do not introduce any modification either at the client or server side of the application. In particular, the PSP module at the mobile host presents a socket-like interface to the application layer.

The PS-Web architecture relies upon estimates of the file transfer times. These estimates are performed by the PS-Daemon at the Access Point and communicated to the mobile host (see [46] for details). As it clearly appears from the protocol description, the accuracy of the estimates is a key factor to achieve a significant power saving at the mobile host. The above architecture can be easily modified to include optimizations like handling of inaccuracies in the estimates supplied by the PS-Daemon (line 5), isolation of the application-dependent functionalities to achieve an higher modularity and reusability, and so on. Details on such optimizations can be found in [46].

```
1: OnNewPageRequested(httpRequest)
2: resumeInterface()
3: send httpRequest to Access Point
4: receive (mainFile,estimate) from Access Point
5: if (estimate $\geq$ MIN_USEFUL_TIME) then
6:     suspendInterface()
7: end if
8: setTimer(estimate)

9: OnRequestFromBrowser(httpRequest)
10: if (interface is ON) then
11:     send httpRequest to Access Point
12: else
13:     insert httpRequest into pendingRequests
14: end if

15: OnTimerExpired()
16: for all httpRequest in pendingRequests do
17:     send httpRequest to Access Point
18: end for

19: OnPageTransferFinished()
20: suspendInterface()
```

```
1: OnNewPageRequested(httpRequest)
2: send httpRequest to server
3: receive mainFile form server
4: estimate = evaluatei_time(mainFile)
5: send (mainFile, estimate) to mobile host

6: OnRequestForEmbedded(httpRequest)
7: file = identifyFile(httpRequest)
8: send file to mobile host
```

Pseudo-code 5.1: PSP protocol: actions performed at the mobile host (left), and at the Access Point (right).

5.4. Power-Saving strategies

In this section we consider the four power-saving strategies mentioned in Section 5.1, i.e., the ideal strategy, the Indirect-TCP strategy, the local strategy and the global strategy (i.e., PS-Web). Specifically, we provide closed form expressions of the energy consumption achieved by the four power-saving strategies in the reference environmnet. This provides analytical tools to better understand their behavior.

In the reference scenario (see Figure 5.3) the legacy TCP/IP protocol stack is typically implemented in the mobile device, and no power saving strategy is used. Therefore, Equation 5.5 instantiates as follows:

$$C_{TCP} = \sum_{i=1}^{N} \left(\frac{D_i + h_i}{\gamma_i\,(TCP)} + \delta_i\,(TCP) \right) + \sum_{i=1}^{N} k_i + t_{UTT}\ . \qquad (5.6)$$

where β_i, $A = \sum_{i=1}^{N} k_i$, $U = t_{UTT}$ and $m = 0$, since no power-saving strategy is used. $\delta_i\,(TCP)$ represents the δ_i term of equation (2) when the legacy TCP/IP architecture is used; and $\gamma_i\,(TCP)$ is the throughput experienced during this transaction.

Several factors contribute to make the legacy TCP/IP approach inefficient from the power-saving standpoint: $\gamma_i\,(TCP)$ is typically very low due to the interaction between the wired and wireless environments [7, 1], U corresponds to the whole User Think Time ($U = t_{UTT}$), and A is the sum of the Active OFF Times ($A = \sum_{i=1}^{N} k_i$). Therefore, C_{TCP} represents the upper bound for the energy consumption. On the contrary, the ideal strategy introduced in the next section represents the lower bound for the energy consumption.

5.4.1. Ideal strategy

The minimum possible energy spent for a Web-page download is obtained by assuming that the transfer from the Access Point to the mobile host is performed in a single phase. Specifically, the wireless interface is turned on, all data are transferred at the maximum throughput allowed by the wireless link, γ, and then the wireless interface remains off until the next Web-page download. Hence, the wireless interface remains on for the minimum amount of time. Accordingly, the ideal energy consumption is given by Equation 5.7.

$$C_{ideal} = \sum_{i=1}^{N} \frac{D_i + h_i}{\gamma} + t_{so} \ . \tag{5.7}$$

Equation 5.7 is immediately obtained from Equation 5.5 by considering that:

- β_i is equal to h_i;
- γ_i is constantly equal to γ;
- there is no temporal overhead related to the HTTP transaction (τ_i=0);
- the Active OFF Times and User Think Time contributions are 0 ($A = U = 0$);
- the wireless interface is turned on only once for each Web-page download (m=1).

It is worthwhile to point out that, even though the ideal strategy is unfeasible, C_{ideal} represents a lower bound for any other feasible power-saving strategy. In the next sections we introduce three feasible power-saving strategies, and compare their performance with the ideal case.

5.4.2. Indirect-TCP strategy

The Indirect-TCP (I-TCP) approach [16] splits the TCP connection (between the mobile client and the remote Web server) in two TCP connections. The former one operates between the mobile client and the Access Point, while the latter one connects the Access Point to the Web server. This allows to decouple the wireless and the wired environments. Hence, the I-TCP approach increases the end-to-end throughput [16], and indirectly contributes to reduce the power consumption. This effect is pointed out by Equation 5.8 that defines the energy consumption related to the I-TCP strategy:

$$C_{I-TCP} = \sum_{i=1}^{N} \left(\frac{D_i + h_i}{\gamma_i\,(I-TCP)} + \delta_i\,(I-TCP) \right) + \sum_{i=1}^{N} k_i + t_{UTT} \ . \tag{5.8}$$

The only difference between Equation 5.8 and Equation 5.6 is $\gamma_i\,(I-TCP)$ instead of $\gamma_i\,(TCP)$, and $\delta_i\,(I-TCP)$ instead of $\delta_i\,(TCP)$. By considering that $\delta_i\,(I-TCP) \approx \delta_i\,(TCP)$ and that the I-TCP approach generally results in an increased throughput (i.e., $\gamma_i\,(I-TCP) \geq \gamma_i\,(TCP)$), Equation 5.8 indicates that $C_{I-TCP} < C_{TCP}$.

A bare I-TCP strategy only provides energy saving as a side effect, since it is not essentially aimed at minimizing energy consumption. In particular, this strategy does not provide any contribution to reduce the second and third terms of Equation 5.6, i.e., A and U. These terms (mainly U) heavily contribute to the energy consumption, since they represent the contributions of idle phases to the energy consumption. To reduce their impact, the wireless interface should remain off as long as possible during idle phases, and hence we expect that a pure I-TCP approach performs poorly. Nevertheless, it constitutes a reference architecture for more efficient strategies. Specifically, in the following we present two strategies that enhance the I-TCP approach. The former one minimizes the contribution of the User Think Time (U) to the energy consumption while the latter one attempts to minimize both A and U. The first strategy only requires information that are local to the mobile host, and hence will be referred throughout as *local strategy*. On the other hand, the second strategy (i.e., PS-Web) needs a global overview of the system, and will thus be referred to as *global strategy*.

5.4.3. Local strategy

This strategy is *local* in the sense that the wireless-interface switching-off decision is taken utilizing only local information. Specifically, the mobile host turns off the wireless interface during the User Think Time. This strategy is very simple to implement. It only requires that the wireless interface is switched off when the Active Phase is finished, and turned on again upon receiving a new request from the mobile user. As this strategy does not modify the I-TCP behavior during the Active Phase, the first and second terms of Equation 5.8 remain unchanged. Furthermore, it eliminates the Inactive Phase contribution (i.e., U=0), and the wireless interface is switched on just once for each Web-page download (i.e., m=1). Hence, the energy consumption using the local strategy is:

$$C_{local} = \sum_{i=1}^{N} \left(\frac{D_i + h_i}{\gamma_i \left(I - TCP \right)} + \delta_i \left(I - TCP \right) \right) + \sum_{i=1}^{N} k_i + t_{so} \ . \tag{5.9}$$

Equation 5.9 is obtained from Equation 5.5 by setting $\beta_i = h_i$, $\tau_i = \delta_i \left(I - TCP \right)$, $A = sum_{i=1}^{N} k_i$, U=0 and m=1.

Since the User Think Time contribution is typically heavy, the energy saving provided by this strategy is expected to be significant. In the next section we will investigate how to further increase the energy saving by switching off the wireless interface even during the Active Phase. A

5.4.4. Global strategy

The global strategy (i.e., PS-Web) attempts to approach the ideal energy consumption by exploiting the knowledge of Web traffic statistics (see Section 5.2.1). The global strategy borrows from the local one the idea of switching off the wireless interface during the Inactive Phase (i.e., U=0). Moreover, it uses the mechanisms described in Section 5.3 to reduce the energy consumption during the Active Phase, as well. Based on the description of the PS-Web architecture and protocols, we are now in the position to proof the following proposition.

Proposition 1 *In a system that adopts the global strategy, the energy consumption is:*

$$C_{global} = sum_{i=1}^{N} \left(\frac{D_i + \beta_i}{\gamma} \right) + A + (s+1) \cdot t_{so} \ . \tag{5.10}$$

Proof. For ease of reading, we move the proof in Appendix A. ■

As shown in the proof of the above proposition (see Appendix A), the global strategy relies upon the algorithm used for estimating the residual transfer-time. Specifically, it requires the estimate of the residual transfer-time for both the HTML main file and the embedded files. The following propositions provide closed formulas for these quantities.

Proposition 2 *By denoting with est_m the residual transfer-time for the HTML main file, the following equation holds:*

$$est_m = \begin{cases} RTT & \textit{if a connection is available} \\ 2 \cdot RTT & otherwise \end{cases} \ . \tag{5.11}$$

Proof. est_m can be evaluated by assuming the knowledge of the RTT between the PS-Daemon and the Web server. When the PS-Daemon receives a request from the mobile host, it establishes a TCP connection with the server, or it uses an already opened persistent connection. In the first case, the retrieval of the main file requires, at least, two RTTs (three-way handshake plus HTTP request-response). In the second case (persistent TCP connection), a single RTT may be enough (if the main file fits into a single TCP?s window size). ■

Proposition 3 *By denoting with est_e the estimate of the residual transfer-time for the embedded files, the following equation holds:*

$$est_e = RTT \cdot u \ , \tag{5.12}$$

where u is the minimum number of RTTs necessary to transfer all the embedded files on a TCP connection.

Proof. The residual-transfer-time estimate for the embedded files exploits some information contained in the main file, that are already downloaded when the embedded files are requested. Hence, the PS-Daemon knows the number, e, of embedded files that compose the Web page[2]. The total number of bytes to be transferred (throughout referred to as B), can be estimated as follows:

$$\hat{B} = \sum_{i=1}^{e} \left(\tilde{D}_i + h_i \right) \ , \tag{5.13}$$

where $\hat{B}$ is the estimate of B, $\tilde{D}_i$ is the estimate for the i-th embedded file size, (i.e., it is a sample from the distribution defined in [19]), and h_i is the dimension of the HTTP headers used for downloading the i-th embedded file. The distribution parameters of embedded-file sizes may vary with the Web servers' content. For this reason they should be communicated by the Web server to

[2]Throughout the analysis, we assume that all the embedded files reside on the same server of the main file. Very similar mechanisms (although slightly more complex) can be used when the embedded files reside on different servers.

the PS-Daemon. This is complex and unrealistic. This complexity can be avoided by using average values only. Accordingly, $\hat{B}$ becomes

$$\hat{B} = e \cdot \left(\bar{D} + \bar{h}\right) \ , \tag{5.14}$$

where $\bar{h}$ is the average of h_i, and $\bar{D}$ is the average of $\tilde{D}_i$. Finally, the residual-transfer-time estimate can be evaluated by using $\hat{B}$ and an estimate of RTT. Specifically,

$$est_e = RTT \cdot u \ , \tag{5.15}$$

where u is the minimum number of RTTs necessary to transfer the bytes on a TCP connection, given the connection state. The complete algorithm to evaluate u is presented in [40], and is omitted here due to space reasons. The estimators of the residual transfer-time require the RTT knowledge. If the PS-Daemon has no information about the RTT, it uses some initial value (as TCP does [53]). ■

5.5. Experimental test-bed

The main objective of our experimental study is to evaluate the power-saving performance of the strategies presented above through an extensive set of measurements on a real Internet test-bed. To this end, we implemented the local and global strategies on top of an I-TCP architecture [16]. In this section we present the performance figures that we intend to investigate, and the characteristics of our test-bed.

5.5.1. Performance indexes

We evaluate the local and the global strategies in terms of energy consumption with respect to a reference I-TCP architecture. Specifically, in the reference I-TCP architecture we assume on the wireless link a transport protocol optimized for the wireless link characteristics, instead of the legacy TCP protocol. This is a light protocol that only implements mechanisms for error detection and recovery and does not include any congestion control mechanism. Hence, an important performance measure is the *power-saving index*, defined as:

$$I_{ps} = \frac{C_{power-savingarchitecture}}{C_{I-TCP}} \ , \tag{5.16}$$

where C_{I-TCP} comes from Equation 5.8 and $C_{power-savingarchitecture}$ is one among C_{ideal}, C_{local} and C_{global} from Equations 5.7-5.10. I_{ps} is the energy consumption of a specific power-saving strategy expressed as a percentage of the energy consumption of the reference architecture. As it will be explained later, our experimental test-bed guarantees that values used to compute I_{ps} are measured under the same system conditions (Web server and network traffic conditions).

In addition, we compare the performance of the local and global strategies with those of the ideal one to understand how well feasible strategies approximate the ideal case.

Although power saving is the key factor to evaluate the proposed strategies, we are also interested to analyze the impact of these strategies on the QoS perceived by the user. Hereafter, we use the URT (i.e., the time interval elapsed from the user request till the rendering of the related Web page) as the main QoS index for a Web service. It is worth noting that the global strategy may introduce an additional delay in the Web-page transfer-time. Additional delays occur whenever a residual-transfer-time estimate is longer than the real value. To take into consideration this aspect, we introduce the *page delay index* defined as:

$$I_{pd} = \text{page transfer-time with global strategy} - \text{page transfer-time with I-TCP strategy}\ . \quad (5.17)$$

I_{pd} measures the additional URT delay introduced by the global strategy.

5.5.2. The test-bed

In our test-bed we use a real Web server located at the University of Texas at Arlington, while the mobile host (and the Access Point) is located at the CNR in Pisa (Italy). This allows us to evaluate the power-saving strategies over a real, congested, intercontinental path. As far as the wireless link, we adopt the Wi-Fi technology with transmission speed ranging from 2 to 11 Mbps.

At the mobile host we use SURGE to simulate a Web client [26, 25, 19]. SURGE reproduces the statistical user model presented in Section 5.2.1. Specifically, SURGE operates in two steps. During the first step, it defines the set of files to be stored in the Web server, guaranteeing that file sizes are distributed as shown in Section 5.2.1. Moreover, SURGE defines the structure of the Web pages by building groups from the above files.

In the second step, SURGE defines the sequence of client requests to the Web server. To this end it creates

- a trace of Web-page requests to be issued to the server (Active Time trace);
- a trace of Inactive OFF Times.

The traffic generated by using the Active Time and Inactive OFF Time traces meets the statistical characterization given in Section 5.2.1.

During the experiments, the client requests and User Think Times are extracted from the above traces. Specifically, a client picks up a Web-page request from the Active Time trace and downloads the corresponding page, then picks up a value from the Inactive OFF Time trace, waits for this time interval, and extracts the next Web-page request.

5.5.3. Experimental methodology

To test our power-saving strategies we ran an extensive set of experiments. In each experiment we have two instances of the same "SURGE client". The two instances download, in parallel, the same set of Web pages by using the pure I-TCP architecture and the selected power-saving strategy, respectively. This guarantees that the two sets of parallel downloads are performed under the same system conditions[3]. For each page download we log the URT value, the total length (in bytes) of the Web transaction (i.e., the sum of the page dimension and the HTTP headers dimension), and the network energy consumption (i.e., the total amount of time during which the wireless interface is turned on). From each experiment, we compute the I_{ps} index for the selected strategy (global, local and ideal). Moreover, for each Web-page download performed with the global strategy we evaluate the I_{pd} index.

A final remark is necessary about the length of each experiment. The Web characterization given in Section 5.2.1 shows that the file size can be modeled according to an hybrid distribution: lognormal in the body and Pareto in the tail. We choose the experiments' length to have – in average – for each experiment at least 10 files' requests coming from the tail of the distribution[4]. From the SURGE documentation, 93% of the requested files comes from the body, while 7% comes from the tail. Therefore, to have (in average) 10 "long" files (i.e., with size belonging to the tail of the distribution), the minimum number of files to be transferred in each experiment is 143. Hence, we decide to stop each experiment after downloading 150 files[5].

We replicated the experiments sequentially throughout an entire working day. To achieve independent experiments, we modified SURGE in such a way that it can start from a specific point in the page requests trace. Exploiting this feature, each experiment starts requesting the trace item next to the last one used in the previous experiment, and hence I_{ps} and I_{pd} samples from different experiments are independent.

It must be noted that an entire working day of experiments is not sufficient to exhaust the whole trace of Web-page requests. Finally, we replicated an entire day of experiments for 10 working days[6].

5.6. Tuning of the experiments

In this section we present some preliminary results collected in the experiments of a single day. These results are used to tune our measurement methodology.

5.6.1. Comparison between the embedded file size estimators

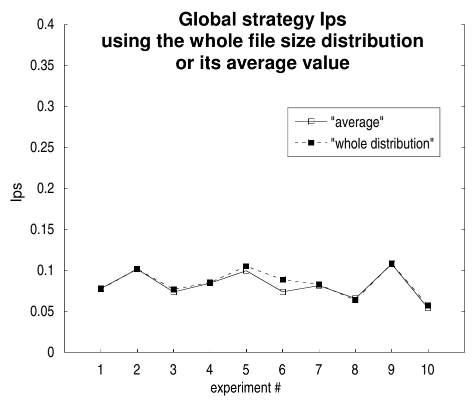
In Section 5.4.4 we described two estimators of the total embedded file size. The first one uses the file size distribution, while the second one relies upon the average value only. To compare these estimators, we ran two sets of 10 consecutive experiments. The two sets of experiments were performed by downloading the same set of Web pages. Furthermore, we verified that all experiments

[3]One of the two users may experience some advantages due to the Web server caching. Specifically, if a user requests the pages immediately before the other one, the latter can find the pages in the server?s file-system cache, and hence it can experience a lower URT. To overcome this asymmetric behavior, the second user starts 30 seconds after the first one; moreover, the user that starts first in an experiment will start as the second in the next one.

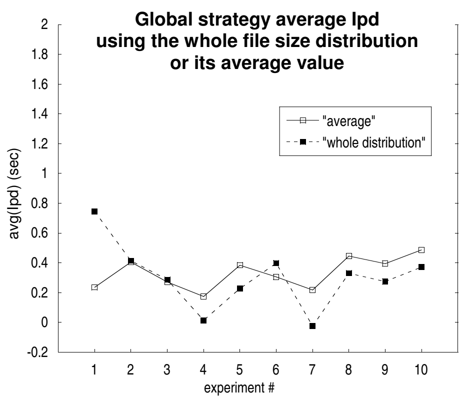
[4]This constraint ensures that results are not biased by a particular choice in the file size dimensions.

[5]The experiment is stopped when the web page "on-the-fly" is completely transferred.

[6]The first experiment of a new day begins with the Web-page request successive (in the trace) to the last one used in the previous day. After the trace is exhausted, SURGE wraps-around and requests the first item.



(a) I_{ps} index



(b) $\overline{I_{pd}}$ index

Figure 5.5.: Embedded file size estimators comparison.

	avg(I_{ps})	**avg($\overline{I_{pd}}$)(sec)**
distribution	0.084±0.012	0.303±0.154
average value	0.081±0.012	0.332±0.075

Table 5.2.: I_{ps} and $\overline{I_{pd}}$ average values and confidence intervals.

were performed under the same network conditions. For each experiment, we measure the I_{ps} index for the global strategy. Moreover, for each page, we evaluate the I_{pd} index, and we average the I_{pd} values on the whole 150-file experiment. Hereafter, $\overline{I_{pd}}$ denotes the averaged value. As the experiments are independent and under the same network conditions, the I_{ps} and $\overline{I_{pd}}$ samples are i.i.d.. The data obtained with the two estimators are presented in Figure 5.5 end in Table 5.2.

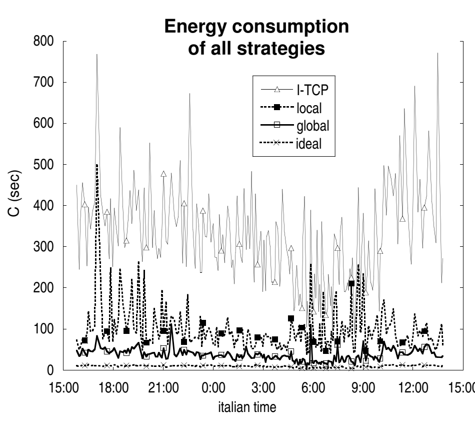
From Figure 4 it appears that the two estimators provide almost identical results in terms of I_{ps} (see part a). In terms of $\overline{I_{pd}}$, the estimator based on the average value appears to be more stable (see part b).

These results are confirmed by Table 5.2, where we report the confidence intervals for I_{ps} and $\overline{I_{pd}}$ (hereafter, the confidence level is 95%). Hence, we can conclude that it is convenient to use the estimator based on the average values.

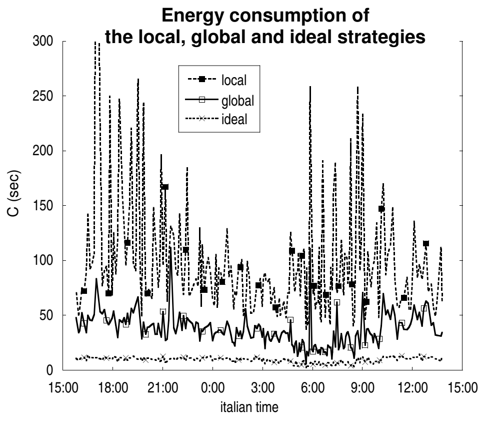
5.6.2. Performance over a single day

To give an idea of the power-saving performance of each strategy, we show some snapshots taken from the experiments of one day.

Figure 5.6(a) shows the plots of C for all strategies under investigation. As expected, the pure I-TCP approach provides the highest power consumption. The gap between C_{I-TCP} and C_{local} is significant, and this confirms that the User Think Time provides a big contribution to the total energy consumption. In Figure 5.6(b)we report the same plots on a different time scale to emphasize



(a) all strategies



(b) ideal, local and global strategies

Figure 5.6.: An experiment day: the energy consumption of the different strategies.

Strategy	*avg*(I_{ps})	*avg*($\overline{I_{pd}}$)*(sec)*
local	0.23	–
global	0.11	0.24
ideal	0.03	–

Table 5.3.: I_{ps} and $\overline{I_{pd}}$ average values for the plots in Figure 5.6.

the differences among the three strategies. These differences are due to the strategies? behavior during the Active Phase. As C_{local} is significantly higher than C_{global}, it follows that a wise energy management during the Active Phase can produce relevant energy savings. Obviously, the ideal strategy is the best one, but the global strategy well approximates the ideal behavior. This observation is confirmed and quantified in Table 5.3.

This table shows the values of I_{ps} and I_{pd} averaged over the whole experiment day. As it clearly appears from Table 3, the global strategy can save 89% of the energy with respect to the I-TCP approach, and it outperforms the local strategy of more than 50%. Furthermore, the global strategy does not introduce a significant QoS degradation. In detail, this strategy increases (in average) the Web-page download time of 0.24 sec that is almost negligible for a Web user.

5.6.3. Data aggregation

From the plots in Figure 5.6 it appears that the energy consumption values (except for the ideal case) are extremely variable during the day. This is mainly due to the variable conditions of the Internet path from Pisa to Arlington. This is confirmed by Figure 5.7, that reports the throughput, measured at the application level, averaged on one-hour intervals.

As expected, the throughput varies during the day. Specifically, we can observe that, from the throughput standpoint, a working day can be subdivided into several classes that depend on the

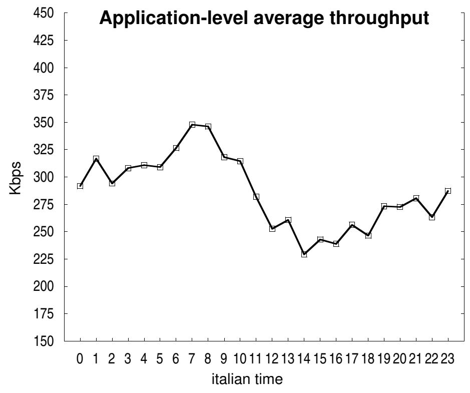


Figure 5.7.: The average throughput measured at the application level as a function of the day-time.

status of the Internet in Europe and USA. For instance, in the period 2PM-7PM we have the minimum throughput due to the overlapping between Europe and USA business hours. On the other hand, in the period 5AM-9AM we observe the highest throughput due to the overlap of non-business hours in Europe and USA. Furthermore, by repeating the same analysis for several working days we observed the same behavior. Thus, hereafter we can assume that experiments performed within the same hour, even in different days, are identically distributed.

Based on the above observations we define our data aggregation method as follows. From each 150-file experiment we derive an observation for I_{ps} and one for $\overline{I_{pd}}$. Our experiments are continuously performed for a whole day and repeated for 10 working days. Samples obtained at the same hour (also in different working days) are i.i.d., and hence from these samples we can compute the hourly confidence intervals of I_{ps} and $\overline{I_{pd}}$, according to the classical statistical method [40].

5.7. Performance evaluation

In this section we deepen the previous analysis by providing, for all strategies, accurate estimates of the confidence intervals of I_{ps} and $\overline{I_{pd}}$ indexes.

5.7.1. Power-Saving Analysis

Figure 5.8 shows the index for the local, global and ideal strategies. The results confirm our preliminary observations. Specifically, by eliminating the power consumption during User Think Times it is possible to achieve a significant energy saving. The local strategy saves about 76% of the I-TCP energy consumption. Moreover, these results also confirm the relevance of the energy management during the Active Phase. The global strategy saves approximately 88% of the I-TCP energy consumption, and therefore significantly improves the local strategy performance.

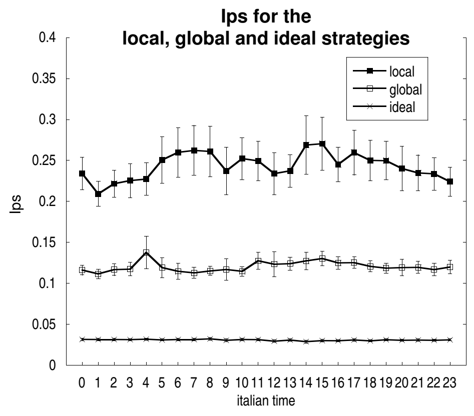
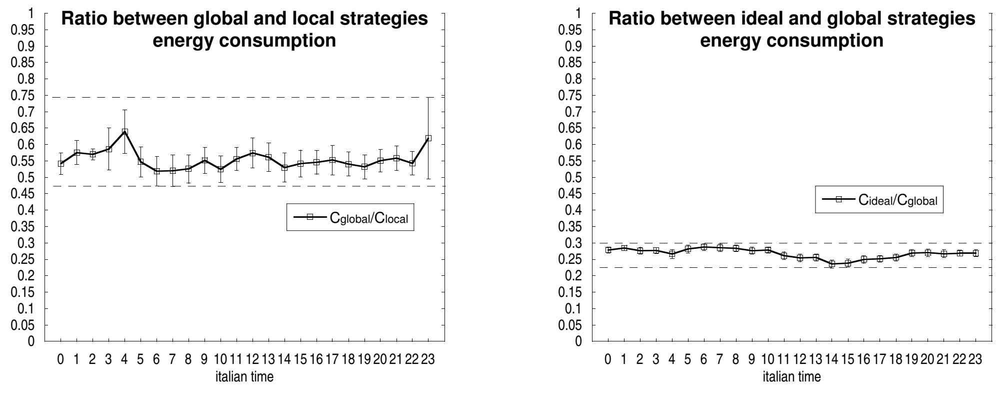


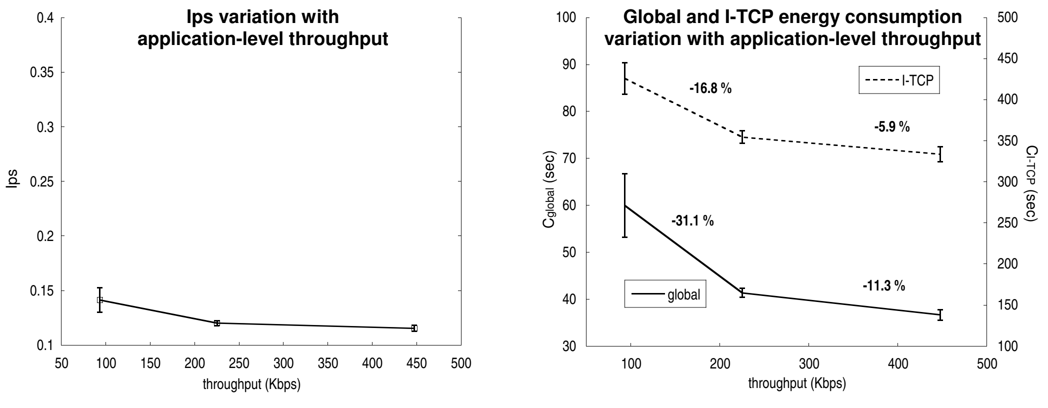
Figure 5.8.: Power-saving performance of the local, global and ideal strategies.



(a) global vs. local

(b) global vs. ideal

Figure 5.9.: Comparison between the local, global and ideal strategies.



(a) I_{ps} (b) C

Figure 5.10.: Analysis of the global strategy I_{ps} as a function of the Internet throughput.

Figure 5.9 compares the local, global and ideal strategies in more detail. Specifically, plot (a) indicates that, with respect to the local strategy, the global strategy saves 26% more energy in the worst case, 45% on the average, and up to 53% in the best case. Plot (b) shows the performance of the global strategy with respect to the ideal case. On average, the saving of the global strategy is approximately 25% of that achievable in the ideal strategy. It is worth noting that, even if not reported here for space reasons, the local and the I-TCP strategies are very far from the ideal case. Specifically, $C_{local} = 7 \cdot C_{ideal}$ and $C_{I-TCP} = 32 \cdot C_{ideal}$.

To summarize, the global strategy is the best approximation of the ideal – *unfeasible* – solution. Therefore, in the following we will focus on the global strategy only. First of all, we analyze the behavior of the I_{ps} index with respect to the throughput on the Internet (γ_i in Equation 5.8). To this end we aggregate the samples in three classes of throughput, and we average the samples belonging to the same class, taking the throughput central value as representative of the entire class. More precisely, classes are made up of samples that experienced a throughput below 150 Kbps, between 150 and 300 Kbps, and between 300 and 600 Kbps, respectively.

Figure 5.10(a) shows that I_{ps} is not very sensitive to the throughput variation. However, it is slightly higher when the Internet throughput is low. This can be easily explained by recalling the residual-transfer-time estimator algorithm. The time interval evaluated as the residual transfer-time estimate is the minimum time to transfer the estimated number of bytes [46]. The choice of the minimum time interval reduces the QoS degradation, but slightly increases the energy consumption when the Internet throughput is low. In this case, the mobile host will need more time to complete the data transfer and, hence, it will switch on the wireless interface several times. This behavior is highlighted by Figure 5.10(b). By moving from the first class to the second one - due to the γ_i increase (see Equation 5.8) - C_{global} decreases of more than 30%, while C_{I-TCP} reduces of 17%. The same trend also occurs in the transition from the second to the third class but the difference is less marked.

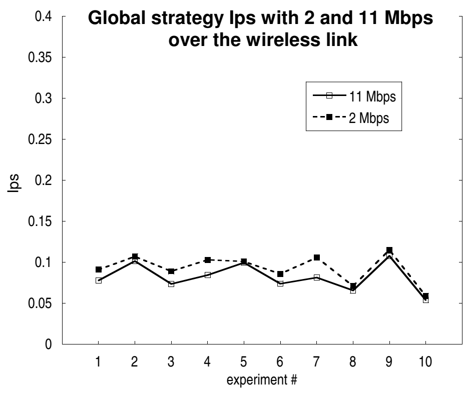


Figure 5.11.: I_{ps} behavior with varying wireless link trhoughput.

Finally, in Figure 5.11 we show the dependence of I_{ps} on the *wireless* link throughput. We ran two sets of ten experiments by varying the speed of the wireless link from 11 Mbps to 2 Mbps. With the current Internet technologies, we expect that the wired Internet remains the bottleneck also when we use a 2Mbps WLAN. As it is clear from Equations 5.8 and 5.10, when the wireless link throughput decreases, C_{global} increases, while C_{I-TCP} doesn't change significantly because it is mainly affected by the wired Internet. However, the results presented in Figure 5.11 show that the global strategy exhibits a small sensitiveness to the throughput of wireless link. By decreasing the wireless speed from 11 Mbps to 2 Mbps, I_{ps} experiences (on average) a 13% increase only.

5.7.2. QoS analysis

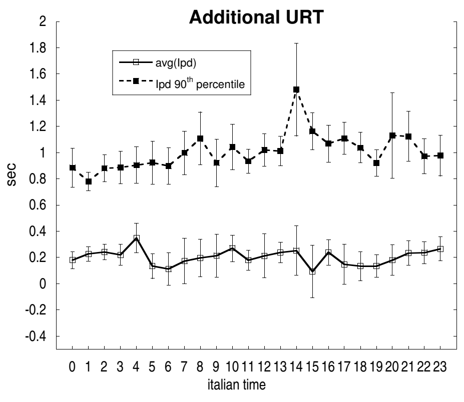
To complete our analysis, we investigate the QoS degradation introduced by the global strategy by studying the I_{pd} index. Specifically, from each experiment we compute the average value of I_{pd} (i.e., $\overline{I_{pd}}$), and its 90th percentile. Then, we average the samples taken within the same hour. Finally, we compute the confidence intervals of the two figures according to the method of Section 5.6.3. The results obtained are shown in Figure 5.12.

From Figure 5.12(a), it can be noted that the additional URT introduced by the global strategy does not degrade significantly the QoS perceived by the Web user: the global strategy increases the URT of about 0.2 sec on average, and no more than 1.8s in the 90% of cases.

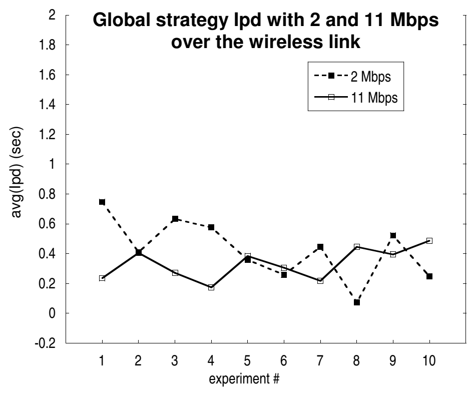
The analysis of the $\overline{Ipd}$ sensitiveness to the wireless link speed confirms the observation done in the I_{ps} analysis (see Figure 5.11). As shown in Figure 5.12(b), by decreasing the wireless link speed from 11Mbps to 2 Mbps the $\overline{I_{pd}}$ index experiences (on average) a 28% increase.

5.8. Summary

We have here proposed and evaluated the effectiveness of new strategies for reducing the power consumption in mobile Web access. Our study starts from the analysis of the impact, on the



(a) $\overline{I_{pd}}$ and I_{pd} 90th percentile average values



(b) $\overline{I_{pd}}$ for different wireless link throghputs

Figure 5.12.: I_{pd} analysis.

power consumption, of the different phases of a Web transaction. To this end we characterize the Web transaction phases through statistical distributions (taken from real Web traffic traces), and construct an energy-consumption model. This model highlights possible directions for reducing the energy consumption. We identify four different power-saving strategies.

We start with a strategy named *ideal* strategy. This strategy guarantees the minimum energy consumption to download a Web page. It is unfeasible but provides a reference for the other strategies we develop. The first feasible strategy we envision is based on a pure I-TCP architecture. The advantage of this strategy is related to the throughput increase (with respect to the legacy TCP/IP architecture) that, indirectly, produces power saving. The second strategy, named *local* strategy, explicitly addresses the power saving. Specifically, by exploiting the semantic of the Web application, it eliminates the waste of power due to User Think Times. The local strategy is furtherly refined by the *global* strategy (also named PS-Web) that performs energy management also during the Active Phases of a Web transaction. The main idea is estimating on-line the length in time of an incoming Web-page download as soon as the Web page is requested by the user at the mobile host. An agent at the Access Point acts as a proxy, and downloads the Web page instead of the mobile host. While the page is "in-flight", the mobile host disconnects, and switches the wireless interface off. Eventually, the mobile host reconnects, downloads the Web page from the Access Point, and renders it in the browser's window. Such a strategy leverages the fact that – typically – the throughput of the Internet path between the Access Point and the fixed host is lower than the throughput of the WLAN (at least, in the cases where few mobile hosts are active in the same hotspot).

For all these strategies we derive an analytical model of their energy consumption, and we compare their performance through extensive measurements. The comparison shows that the local strategy saves, on average, 76% of the energy drained by a pure I-TCP solution. The global strategy outperforms the local one by reducing the I-TCP energy consumption of about 88% on average. In addition, with respect to the ideal strategy, the global strategy consumes less than 4 times, while the local and the I-TCP strategies consume 7 and 32 times, respectively. Therefore, among the analyzed strategies, the global one is the best approximation of the ideal – but unfeasible – case. Furthermore, the global strategy introduces a negligible degradation in the QoS perceived by the users. Specifically, the additional URT introduced by the global strategy is about 0.2 sec on average, and is below 1.8 sec in the 90% of cases. Finally, a sensitiveness analysis shows that the performance of the global strategy is almost independent from the throughput of the wireless link, provided that it is greater than the throughput available in the wired Internet.

6. Application-Independent Power Management

6.1. Overview

In this chapter we design and evaluate a power-saving solution (*PS-WiFi*) that exploits the application-independent approach in the sense that envisaged strategies do not exploit knowledge about the above applications. Our solution presents to the above layer a standard socket interface, and thus it does not require any modification in the applications. In addition, it is completely independent from the sub-network technology. PS-WiFi is complementary to PS-Web presented in the previous part, which exploits the application-dependent approach.

We implemented the application-independent solution in a prototype system and extensively tested it. The target of our experiments was twofold: i) to understand how our solution performs in an actual Internet scenario, with respect to the power savings, and the QoS perceived by the user; and ii) to compare and contrast it with the application dependent solution. Our experimental results indicated that both the dependent and the independent approach guarantee a significant power saving: the application dependent solution saved, on average, around 90% of the legacy TCP/IP-architecture energy- consumption, while, in the same application scenario and under similar conditions, the application independent solution saved, on average, around 80% of the legacy TCP/IP-architecture energy-consumption. Furthermore, these reductions in the power consumption were obtained without a significant degradation of the QoS. Specifically, we measured the increase in the User Response Time (URT) caused by our power saving architectures. With respect to the legacy TCP/IP-architecture, the application dependent approach increased the URT of less than 2 seconds, while in the application independent approach the additional URT was less than 2.5 seconds.

Furthermore, we deepen the previous experimental analysis by considering e-mail traffic in addition to Web traffic. We also analyze the performance of the proposed solution in the presence of multiple concurrent applications (e.g., e-mail and Web). The aim of this analysis is to show that the proposed application-independent policy exhibits good performance irrespectively of the network application(s), and is thus suitable for a real environment. The experimental results show that our application-independent policy is both flexible and efficient.

However, as the measurement study was carried out using the real Internet, the environment parameters were not completely under our control. Hence, these measurements reported do not allow analyzing the performance of PS-WiFi extensively. To better understand the potentialities of the PS-WiFi approach we develop an analytical model of PS-WiFi. For the sake of simplicity, we considered the Web browsing as the reference application. The first step of our study is the

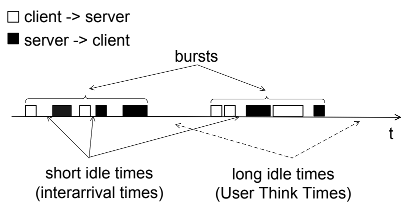


Figure 6.1.: Snapshot of a typical best-effort data exchange.

definition of a traffic model for a typical Web user. This model is then exploited to provide closed formulas that describe the performance indexes of PS-WiFi under this load conditions.

Our model is very accurate. The difference between the analytical results and the measurements on the real testbed is, on average, about 1% for the energy-saving performance index, and about 7% for the additional transfer-time. Therefore, the analytical model is a flexible tool to analyze the behavior of PS-WiFi. We used this model to better understand: (i) which parameters determine the performance of PS-WiFi; and (ii) how parameter values affect the PS-WiFi performance. Specifically, we analyze the sensitiveness of the system with respect to the main Internet parameters, i.e., the throughput and the Round Trip Time (referred to as RTT) between the Web client and the Web server. The results show that power saving is mainly affected by throughput variations. Specifically, power saving varies from 48% to 83% when the throughput increases from 0 to ∞. However, for typical throughput values (i.e., between 50 Kbps and 1 Mbps) variations of power saving are limited (between 68% and 82%). On the other hand, the additional transfer-time is a slightly increasing function of RTT and never affects significantly the QoS perceived by Web users, since the average additional transfer-time is always less than 0.5 sec.

6.2. The PS-WiFi system

6.2.1. Power-saving management of best-effort traffic

Our power-saving architecture was designed to support any network applications with the following traffic pattern: data-transfer phases are characterized by *traffic bursts* interleaved by *idle phases* (during which data are processed locally). During each burst several packets are exchanged. Packets inside a burst are separated by *short idle times*, while consecutive bursts are separated by *long idle times*. Short idle times and long idle times are generated by different phenomena. Short idle times are driven by network protocols and data processing (e.g., a TCP-sender waiting for receiving acks after sending segments), while long idle times are related to human times (e.g., a new Web page is requested after the user has read the previous one). Due to their different nature, short idle times are much smaller than long idle times, and 1 sec is typically assumed as the cut-off value between the two classes. Figure 6.1 shows a snapshot of a typical data exchange. It should be noted that idle times are measured assuming the mobile device perspective: an idle time starts whenever the mobile device has no more packets to exchange, and finishes when the mobile device sends or receives a new packet.

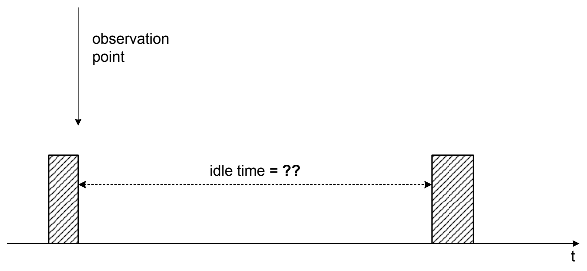


Figure 6.2.: PSA scenario.

The ideal power management would switch off the wireless interface during idle times and resume it whenever a new packet is ready to be exchanged. Two factors make this ideal policy unfeasible in practice. First, the duration of an idle time – i.e., the instant in time when the wireless interface must be resumed – is unknown a priori. Second, the wireless interface has a switching-on transient time (referred to as t_{so}) during which it consumes energy but can not communicate. Hence, for idle times less than t_{so}, it is energetically convenient to leave the network interface on. To overcome these challenges, the PS-WiFi approach is based on a dynamic estimate of the duration of idle times. Specifically, i) we measure at run time the duration of idle times; ii) we use these measurements to predict the length of the next idle time; and iii) we use this next idle-time prediction to decide whether the wireless interface should be switched off of not. If the wireless interface is switched off, the next idle-time prediction is also used to decide when to resume it. Therefore, the core component of PS-WiFi is a set of algorithms for predicting the duration of the next idle time.

As short and long idle times are generated by different phenomena, PS-WiFi includes two distinct algorithms for estimating them. Short idle times are estimated by means of the Variable-share Update Algorithm (VUA) [33]. On the other hand, long idle times are estimated by means of a binary exponential backoff policy (see below). It must be noted that the estimator accuracy is very important for short idle times, while it is less crucial for long idle times. Long idle times typically finish when the user of the mobile device sends a message to the fixed host. Hence, the first packet after a long idle time is sent by the mobile device to the fixed host (see Figure 6.1). Therefore, overestimates of the long idle times have no impact on performance since the mobile device resumes the wireless interface as soon as a new packet is generated. On the other hand, packets inside a burst also travel on the opposite direction (i.e, from the fixed host to the mobile device). In this case, if the mobile device switches the wireless interface off it is not aware when packets coming from the fixed host arrive and becomes aware only when it resumes the wireless interface, after the *estimated* idle time has elapsed (see the next section for details). Therefore, an overestimate may increase the transfer time experienced by that packet. For these reasons, PS-WiFi includes a very accurate estimator for short idle times (VUA), and a simpler estimator for long idle times.

Based on these two estimators, we have designed a Power Saving Algorithm (PSA). Let us assume that idle times are detected by PS-WiFi as soon as they start. Then, the question to answer is: “How can we estimate the length of that idle time?” (Figure 6.2 gives a graphical representation of this scenario). When an idle time occurs, we have just received a packet, and hence we are likely to be inside a burst. Thus, a *short* idle time is assumed, and VUA provides an estimate, say t', of the (supposed) short idle time. If this estimate occurs to be too short, an update must be provided[1]. To this end, by exploiting the distribution of short idle times estimated from the history, the 90^{th} percentile (throughout referred to as k) is used as the updated estimate. As t' seconds have already elapsed, the new packet is expected within the next $k - t'$ seconds. If a new packet is still not available after this time interval, the idle time is greater than k seconds. This means that the probability of the idle time being a short idle time is below 10%, and hence a *long* idle time is assumed. The estimate is set to the minimum possible value of long idle times[2], i.e., 1 sec. As k seconds have already elapsed, the new packet is expected within the next $1 - k$ seconds. If needed, further updates are generated using a binary exponential backoff procedure[3], until the long idle time finishes.

When the idle time ends, if it was a short idle time, then its value is provided to VUA to update its parameters.

To summarize, PSA exploits both estimators of short and long idle times. If $u^{(i)}$ denotes the sequence of estimates provided by PSA when an idle time occurs, and $z^{(i)}$ denotes the corresponding sequences of intervals within which new packets are expected (see Figure 6.3), the following equations hold:

$$\begin{cases} u^{(0)} = t' \\ u^{(1)} = k \\ u^{(2)} = 1 \\ u^{(3)} = 2 \\ u^{(4)} = 4 \\ \dots \\ u^{(n)} = 2^{n-2} \end{cases}, \quad \begin{cases} z^{(0)} = t' \\ z^{(1)} = k - t' \\ z^{(2)} = 1 - k \\ z^{(3)} = 1 \\ z^{(4)} = 2 \\ \dots \\ z^{(n)} = u^{(n)} - u^{(n-1)} \end{cases} . \qquad (6.1)$$

Specifically, $u^{(0)}$ and $u^{(1)}$ are provided by VUA, while $u^{(2)}$, ..., $u^{(n)}$ are provided by the long idle-time estimator. It is worthwhile noting that PSA is memory-less, in the sense that both $u^{(i)}$ and $z^{(i)}$ related to a particular idle time are independent of sequences related to previous idle times.

Finally, it should be noted that the assumption that idle times are detected by PS-WiFi as soon as they start may not hold. For example, due to an overestimate of the previous idle times, the PSA may be executed when the idle time has already started. We will explain how PSA operates in this case.

At this point, is is worth explaining how VUA works in the PS-WiFi system. Specifically, the following section is devoted to this.

6.2.2. Algorithm for packet arrival estimates

As clearly appears from the previous section, our solution relies upon the prediction of the traffic behavior. Therefore, we need an algorithm that provides accurate estimates of packet inter-arrivals

[1]The way PS-WiFi detects that an estimate is too short depends on the implementation in the network architecture, and hence it is discussed in the next section.
[2]1 sec corresponds also to the upper bound of the short idle times.
[3]Updates are equal to 2^k, with $k = 1, 2, 3 \dots$.

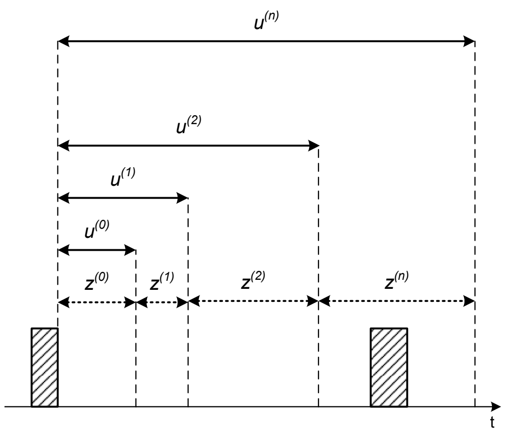


Figure 6.3.: Sequence of updates provided by PSA to estimate an idle time.

times, and is able to adapt quickly to changes in the traffic conditions. The Variable-Share Update algorithm [33] fits these requirements. This algorithm has been proposed as a dynamic algorithm to estimate a generic variable spanning a given range, and is not bound to a specific problem.

Let I be the range of possible values for a variable y that we want to estimate. To predict the value of y, the Variable-Share Update algorithm relies upon a set of "experts". Each expert x_i provides a (fixed) value within the range I, i.e., a value that y can assume. The number of experts to be used, as well as their distribution among the range I, are input data for the algorithm. Each expert x_i is associated with a weight w_i, a real number that measures the dependability of the expert (i.e., how accurately the expert has estimated y in the past). At a given time instant, an estimate of y is achieved as the weighted sum of all experts, using the current w_i value as the weight (i.e., reliability) for the expert x_i. Once the actual value of the variable y is known, it is compared with the estimates provided by the experts, to recalculate and update the weight associated with each expert.

The algorithm is summarized in Figure 6.4. As shown in the figure, the core of the algorithm is the weights updating algorithm. Updates occur every time a new actual value of the variable y becomes available. First, an error function L is evaluated for each expert: this function measures the deviation of the (prediction provided by the) expert from the actual variable's value. Then, the Variable-Share Update is executed, as follows:

1. each expert loses a portion of its weight, according to the deviation from the actual value; the weight w_i becomes w'_i (if L=0 the weight is not changed);

2. each expert shares a portion of its weight, according to its error function: a pool is created by using all the shares (if L=0 the expert doesn't share anything);

3. for each expert, the new weight is calculated as the sum of two components: a portion of the weight evaluated in 1, and a portion of the pool evaluated in 2. Both components depend on the error function (e.g., if L=0, the new weight is the old one, plus a fraction of the pool).

Parameters:	$\eta > 0$, $0 \leq \alpha \leq 1$, n (number of experts)
Variables:	x_i (experts), w_i (weights), y (actual variable's value), $\hat{y}$ (estimated variable's value)
Initialization:	$w_i = 1/n, \forall i = 1, \ldots, n$
Prediction:	$\hat{y} = \sum_{i=1}^{n} w_i x_i / \sum_{i=1}^{n} w_i$
Loss Update:	$w_i' \triangleq w_i e^{-\eta L(y,x_i)}$
Variable Share:	$\begin{cases} pool = \sum_i \left\{1-(1-\alpha)^{L(y,x_i)}\right\} \cdot w_i' \\ w_i = (1-\alpha)^{L(y,x_i)} w_i' + \frac{1}{n-1} \cdot \left\{pool - \left[1-(1-\alpha)^{L(y,x_i)}\right] \cdot w_i'\right\} \end{cases}$

Figure 6.4.: The Variable-Share Update algorithm.

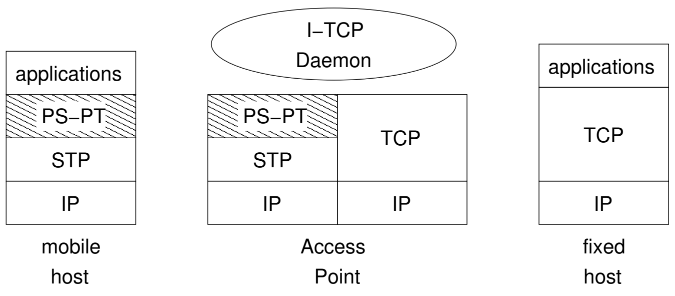


Figure 6.5.: PS-WiFi network architecture (evidence on added components).

The Variable-Share Update algorithm reduces the weights of those experts that provides bad predictions, and increases the weights of the experts that provide the more accurate predictions. The speed in increasing/decreasing weights is determined by two algorithm parameters: α and η. This algorithm has been proposed to implement a spindown technique in hard disks power management [32]. In that context, the update policy guarantees a quick adaptation to changes of the variable's values.

We used 20 experts for each set. The experts values are uniformly distributed over the corresponding intervals (experts of the first set are placed every 50 msec, while experts of the second set are spaced by approximately 3 seconds each other). The update policy shown in Figure 6.4 needs two additional parameters, α and η. According to [32][4], we used α=0.08 and η=4. Finally, the algorithm requires the definition of a loss function L. This function provides a measure of the deviation of each expert from the actual value of the variable, and its values must lie in [0,1], see [33]. In our implementation we used $L(y, x_i) = |x_i - y| / \max_i |x_i - y|$ as the error function.

6.2.3. Network architecture and protocols

To integrate PSA in the Wi-Fi hotspot scenario, we define the network architecture shown in Figure 6.5.

We implement the PSA in the *Power-Saving Packet Transfer* (PS-PT) protocol. As a design choice, the PSA is completely executed at the Access Point. Hence, the impact of the power-saving system on the mobile-device computing resources is negligible. The PS-PT protocol was implemented as a simple master/slave protocol. When there are no more data to be exchanged, the Access

[4] [32] compares several values for α and η assuming a uniform distribution of experts. The experimental results indicate α=0.08 and η=4 as the best choice.

```
 1: OnPacketFromApplications(packet)
 2: timestamp(packet)
 3: if card is OFF then
 4:    stop timer
 5:    turn card ON
 6: end if
 7: send packet to Access Point

 8: OnPacketFromAccessPoint(packet)
 9: cmd = extract_command(packet)
10: if cmd = OFF then
11:    turn card OFF
12:    $t_i$ = extract_interval(packet)
13:    set_timer($t_i$)
14: end if

15: OnTimerExpired()
16: turn card ON
17: send ON_signal to Access Point
```

```
 1: OnNewPacket(packet)
 2: if card is OFF then
 3:    timestamp(packet)
 4:    buffer(packet)
 5: else
 6:    stop timer
 7:    send/receive data
 8:    $t_i$ = evaluate_next_interarrival()
 9:    if card must get OFF then
10:       send (OFF_CMD, $t_i - t_{so}$) to Mobile Host
11:    else
12:       set_timer($t_i$)
13:    end if
14: end if

15: OnTimerExpired()
16: $t_i$ = update_estimate()
17: if card must get OFF then
18:    send (OFF_CMD, $t_i - t_{so}$) to Mobile Host
19: else
20:    set_timer($t_i$)
21: end if

22: OnMobileGetsON()
23: if there is nodata to exchange then
24:    $t_i$ = update_estimate()
25:    if card must get OFF then
26:       send (OFF_CMD, $t_i - t_{so}$) to Mobile Host
27:    else
28:       set_timer($t_i$)
29:    end if
30: else
31:    send/receive data
32:    $t_i$ = evaluate_next_interarrival()
33:    if card must get OFF then
34:       send (OFF_CMD, $t_i - t_{so}$) to Mobile Host
35:    else
36:       set_timer($t_i$)
37:    end if
38: end if
```

Pseudo-code 6.1: PS-PT protocol: actions performed at the mobile host (left), and at the Access Point (right).

Point decide whether it is convenient to the mobile host to switch the network interface off. If so, it sends a "shutdown" command to the mobile host including an indication of the time interval during which the mobile host should remain disconnected. The mobile host uses this interval to set a timer. Upon the timer expiration, the mobile host polls the Access Point again. In the following, we shall describe in detail the actions performed at the mobile host, and at the Access Point, respectively (the pseudo-code here is optimized for clarity rather than for efficiency).

Upon reception of a new packet from the above application(s), the mobile host bounds a timestamp to the packet (this timestamp will be used at the Access Point to maintain the history of the arrival times, see line 2). If the network interface is OFF, then the timer used to signal when the mobile host have to reconnect and poll the Access Point is active. The mobile host stops this timer (i.e., the last estimate was too large, lines 3-6) and sends the packet to the Access Point (line 7). When a new packet from the Access Point arrives, the mobile host checks whether it contains a shutdown command (lines 9-10). In this case the packet also includes the time interval during which the network interface should remain OFF. The mobile host switches the network interface OFF, and sets the timer accordingly (lines 11-14). Finally, upon timer expiration, the mobile host polls the Access Point (lines 15-17).

At the Access Point side, the system records the state of the mobile host's network interface. Upon reception of a new packet (from the Internet) while the mobile host is disconnected, the Access Point buffers the packet and waits for a poll from the mobile host (lines 2-4). On the other hand, if the packet is received while the mobile host is connected, the Access Point relays the packet to the mobile host (if it was received from the Internet, line 7), estimates the next packet arrival time (i.e., $u^{(0)}$, line 8), and decides whether its is convenient to shut down the network interface (lines 9-13). It is worthwhile to recall that the network interface has a transient period t_{so} in getting on during which it is not able to handle data. This implies that the mobile host must be ON t_{so} units of time before the estimated arrival (line 10). If the Access Point estimates that it is convenient for the mobile host to disconnect (i.e., $u^{(0)}$ is greater than t_{so}, it sends a OFF command to the mobile host together with the time interval during which it must remain disconnected (lines 9-10). Otherwise, it sets a timer with the estimated arrival time (lines 11-12). In the latter case the mobile host remains connected. Therefore, if a new packet arrives, the network interface is ON and, hence, the system must stop the timer (line 6).

When the mobile host polls the Access Point, there might be data to exchange or not. In the former case, the Access Point uses the new data to generate a new estimate and performs the same actions described above (lines 30-38). In the latter case, the last estimate provided to the mobile host was too short. Thus, the Access Point updates this estimate (i.e., generates $u^{(1)}$ or $u^{(2)}$ and so on) and decides what the mobile host must do (i.e., checks whether $z^{(i)}$ is greater than t_{so} or not, lines 23-30). The same situation occurs when the timer expires: the last estimate was too short, but it didn t cause the switching off of the network interface. The mobile host is still connected and the Access Point has to decide whether it is convenient that the mobile host disconnects or not (lines 15-21).

6.2.4. Measuring idle times in PS-WiFi

To complete the description of PS-WiFi we discuss how idle times are measured. For ease of understanding we assumed so far (see Section 6.2.1) that idle times are measured at the mobile device. However, measuring idle times at the mobile device is not always the right choice. Let us focus on the example presented in Figure 6.6 (for simplicity in the figure we neglect transfer times on the WLAN; in addition, time intervals during which the wireless interface is on are defined by PSA). Due to possible overestimation, packets A and B may be delayed at the Access Point and, hence, the idle time between A and B measured at the mobile device is affected by the estimate error. In the ideal case, PS-WiFi should be transparent to the traffic generated by the applications, i.e., it should not modify idle times with respect to the case when no power management is used. Hence, idle-time measures should not include additional components introduced by PS-WiFi. Thus, idle times between consecutive packets in the downlink direction (e.g., A and B in Figure 6.6) are measured at the Access Point. On the other hand, consecutive packets in the uplink direction (e.g., C and D in Figure 6.6) are measured at the mobile device. Indeed, t_{so} seconds are added by PS-WiFi to that idle times, if the second packet (i.e., D) is generated when the wireless interface is off. Measuring idle times between packets flowing in opposite directions (e.g., B and C in Figure 6.6) requires a mixed approach. To clarify this concept, let us focus on packets B

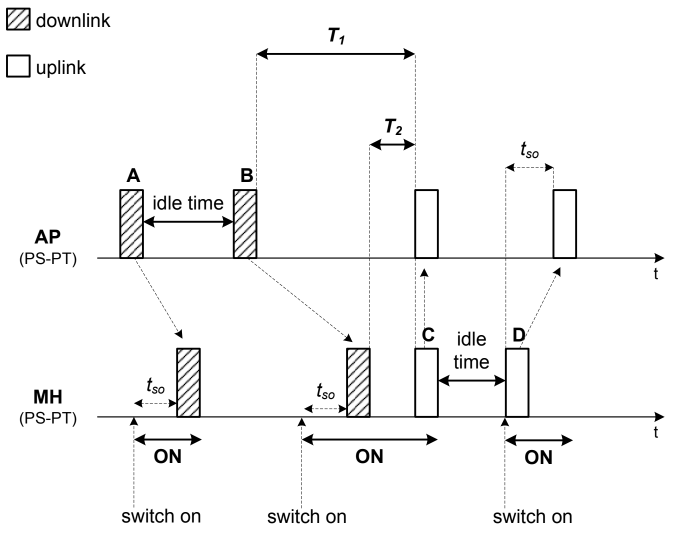


Figure 6.6.: Example of packets exchange in a PS-WiFi system (focus on idle-time measurements).

and C in Figure 6.6. If we assume that C is generated independently by B (e.g., B and C are related to different applications running at the mobile device concurrently), then the idle time to be measured is T_1. However, the Access Point might measure $T_1 + t_{so}$ if the wireless interface was off when C was generated, while the mobile device would measure T_2. The right idle-time value could be measured by i) synchronizing the mobile device with the Access Point[5]; ii) recording the time instant when B arrived at the Access Point; and iii) recording the time instant when C arrived at the mobile device. It must be noted that this strategy is no longer correct if C is generated by the mobile device in response to B, since in this case the right idle time would be T_2. However, typical applications used in WiFi hotspot (e.g., Web browsing, e-mail, file transfer) generate a traffic that is *almost mono-directional*, and, hence, consecutive packets related to the same application flowing in opposite directions can be considered as an exception. To summarize, idle times are measured by PS-WiFi as follows: i) the mobile device and the Access Point are assumed to be synchronized, ii) the instant in time when each packet arrives at the PS-PT layer is recorded, and iii) idle times are measured as the difference between those time instants related to consecutive packets.

The strategy used to measure idle times also impacts on the way idle-time estimates are used. Specifically, since idle times are seen as inter-arrival times, idle-time estimates must be considered accordingly. Let us focus on Figure 6.7, where u_j denotes the final update provided by PS-WiFi for the $j-\text{th}$ idle time (u_j is an item of the sequence $u^{(i)}$ shown in Equation 6.1). Packet A is sent to the mobile device when the estimate u_0 has elapsed. After A has been sent (point K in the figure) no more data is available and, hence, PSA is invoked. As idle times are inter-arrival times measured at the PS-PT layer, estimated idle times must start at the point in time when the previous packet (A in our example) has arrived at the PS-PT layer. Therefore, in our example, the estimated idle time (i.e., u_1) starts at time H, though the estimate is generated at time K. It

[5]Please note that in a 802.11 WLAN the MAC layer requires synchronization and, hence, synchronizing the PS-PT layer has no cost.

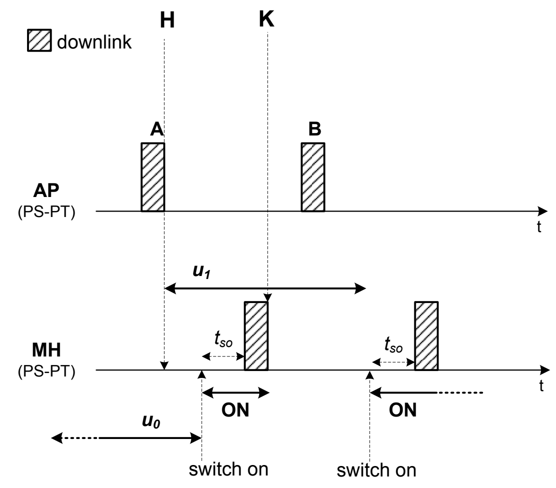


Figure 6.7.: Use case of idle-time estimates.

should be noted that PSA may update the first estimate related to an idle time before sending it to the mobile device. The Access Point exploits the knowledge that the idle time is at least $K - H$. Hence, when packet A is sent, PSA considers the first item in the sequence $u^{(i)}$ that is greater than $K - H$, and sends this estimate to the mobile device along with packet A (see Figure 6.7)[6].

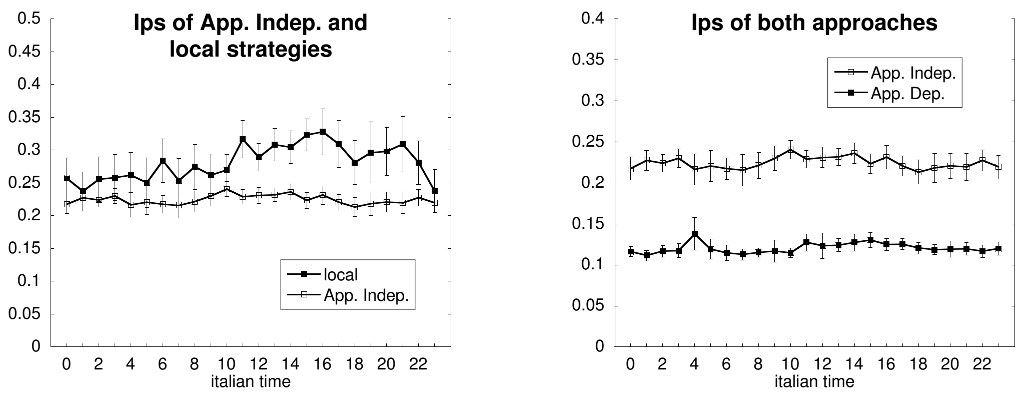
As a final remark, it is worth noting that PS-WiFi approximates the ideal power-saving strategy for the wireless interface: i) data transfers on the wireless link occur at the maximum available throughput on it and are not affected by the throughput on the (wired) Internet; and ii) the wireless interface remains switched off during idle periods.

6.3. Experimental Analysis

In this section we compare the performance of our energy saving architecture with that of an Indirect-TCP architecture using the STP protocol over the wireless link (this architecture is throughout referred to as legacy architecture). We considered Web and e-mail as testing applications as they are today the most popular Internet applications. Furthermore, they both are somewhat sensitive to delays. Thus, it is important to provide not only a significant energy saving but, also, an acceptable QoS level (i.e., to minimize the additional delay introduced by energy management).

As far as the performance indexes, we use the I_{ps} and I_{pd} indexes defined by Equations 5.16 and 5.17. Actually, in the case of the e-mail traffic, the I_{pd} index should be slightly modified. Specifically, in that case it is defined as the additional delay introduced by PS-WiFi to an "e-mail check". An e-mail check occurs when the user on the mobile host polls the POP3 server for new mail (in our model, the user sends queued messages – if any – to an SMTP server after checking her POP3 mailbox).

[6]Clearly, the estimate is sent only if the residual (estimated) idle time is greater than t_{so}.



(a) PS-WiFi vs. local

(b) PS-WiFi vs. PS-Web

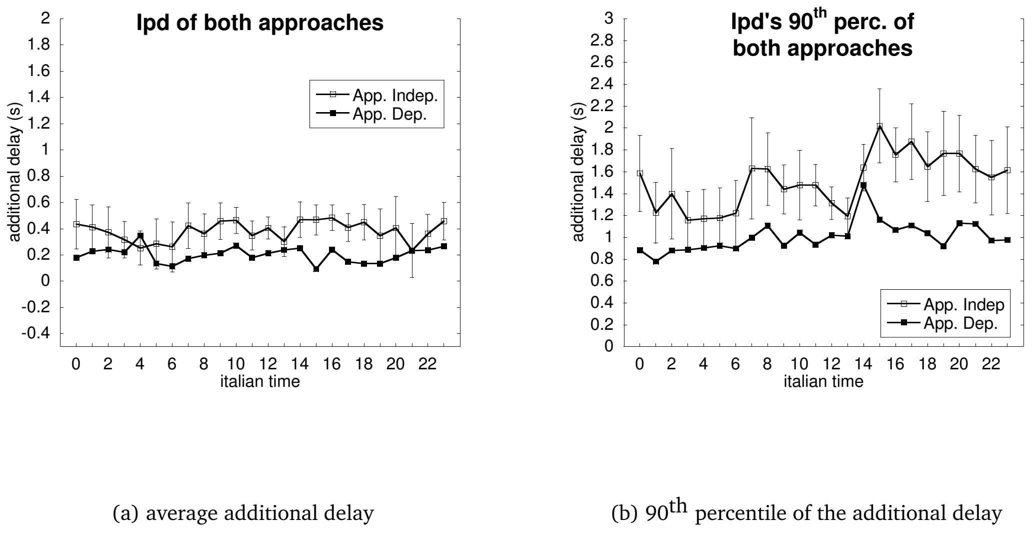
Figure 6.8.: PS-WiFi in the Web scenario: power-saving performance.

To test the flexibility of our solution, i.e., its ability to adapt to different traffic profiles, we considered three different traffic scenarios. In the first scenario we assumed that Web browsing is the only active application. In the second scenario we considered e-mail instead of Web. Finally, in the third scenario Web and e-mail are assumed to be simultaneously active.

6.3.1. Scenario I: Web traffic

In this scenario Web browsing is the only active network application and, hence, the MH generates a single type of traffic. In our experiments we considered a real Web server located at the University of Texas at Arlington, while we used SURGE [19] to simulate the application layer at the client side. SURGE is a Web traffic simulator that reproduces the statistical properties of traffic generated by a realistic Web user. The client was located at the Department of Information Engineering of the University of Pisa (Italy). Hence, our client-server path crossed (congested) intercontinental links, and this allowed us to test our energy management policy in a congested situation.

We performed a large number of experiments. Each experiment included 100 page-transfer operations from the Web server to the client (an experiment stopped when the whole page "in flight" arrived at the client). In each experiment, the same set of pages were requested in parallel both in our architecture, and in the legacy architecture. This guarantees the same network conditions in both cases. We ran a set of experiments spanning an entire working day. Furthermore, to increase results' reliability, we replicated the experiments in several working days. Throughout, we present average hourly values and the related confidence intervals (confidence level 95%). Figure 6.8(a) shows the I_{ps} index as a function of time. It clearly emerges that our architecture allows



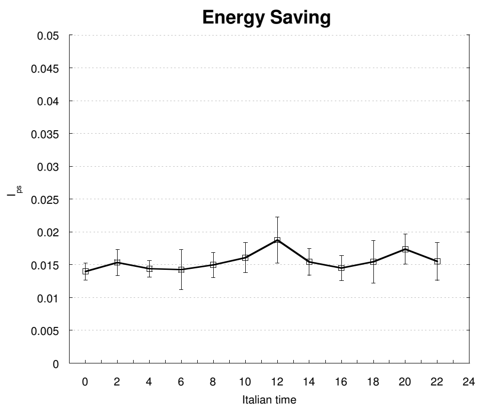
(a) average additional delay

(b) 90th percentile of the additional delay

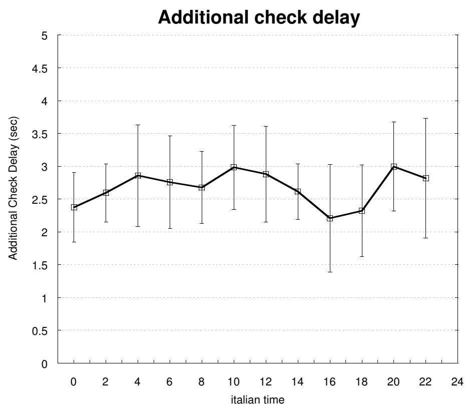
Figure 6.9.: PS-WiFi in the Web scenario: QoS performance.

a significant energy saving with respect to the legacy architecture. The energy saving achieved is in the order of 78%. For comparison, Figure 3 (a) also shows the I_{ps} index related to the *local* policy (see Section 5.2). It is worth recallint that this strategy switches off the wireless interface when the Web page download is complete, and resumes it upon receiving a new request from the user (i.e., it saves energy only during Think Times). This strategy is called "local" because it can be implemented by only exploiting information available locally at the Web browser. Unlike the applicationindependent policy, the local policy depends on the particular application we are considering, i.e., it is application dependent. From the comparison of the two curves it emerges that our application-independent policy performs better than the local policy. This means that it saves energy even during the page-download phase. Figure 6.8(b) compares PS-WiFi and PS-Web. As expected, PS-Web performs the best, since it exploits a-priori knowledge about the Web-user behavior. However, the performance degradation of PS-WiFi is not very high.

Figure 6.9(a) shows the average additional delay introduced in downloading a Web page with respect to the legacy architecture. The additional delay introduced by the PS-WiFi is very low. In our experiments the average value is in the order of 0.4 sec, while the 90th percentile is typically below 2 sec, and always below 2.5 sec (Figure 6.9(b)). Also in this case, PS-Web performs better than PS-WiFi, but the performance degradation of PS-WiFi is tolerable. Based on the above results we can conclude that the QoS degradation introduced by the PS-WiFi policy can be considered as acceptable for Web-browsing applications.



(a) I_{ps}



(b) I_{pd}

Figure 6.10.: PS-WiFi in the E-mail scenario.

6.3.2. Scenario II: E-mail traffic

In the second scenario we consider e-mail instead of Web. Along with the previous one, this scenario is aimed at showing that the application-independent policy exhibits good performance when there is a single running application, irrespectively of the specific application.

E-mail involves two protocols: POP3 for downloading messages from the POP server to the user s computer (the MH in our case), and SMTP for sending messages from the user s computer to the SMTP server that, in its turn, will forward the same messages to the final destination. To perform our experiments we developed application programs that simulates the statistical behavior of a POP3 server, an SMTP server and an e-mail client (POP3 + SMTP), respectively. The statistical behavior of these programs was derived from previous experimental studies [20]. Since we are considering a mobile environment, in our experiments we assumed that the user is outside his/her home location. This implies that the POP3 and SMTP servers do not belong to the same Local Area Network (LAN) of the AP. The alternative scenario (i.e., POP3 and SMTP servers belonging to the same LAN of the AP) is less meaningful since, in such a scenario, the wireless link is very well exploited and energy management becomes almost useless. We assume that the user periodically connects to the (remote) mail server for sending and/or receiving e-mail messages (if any). The time interval between two consecutive checks was assumed to be 5 minutes (in order to have a significant number of checks for each experiment).

Figure 6.10(a) shows the I_{ps} index as a function of time for the application-dependent and local policies, respectively. Figure6.10(b) reports the average additional delay introduced by the two policies for each e-mail check. From the comparison it emerges that the two policies exhibit similar performance in terms of energy saving (they both save about 85% of the energy consumed in the legacy architecture). This is because the period between two consecutive e-mail checks (5 minutes in this case) is largely predominant with respect to the datatransfer time (typically in the order of seconds). Therefore, reducing energy consumption even in the datatransfer phase does not

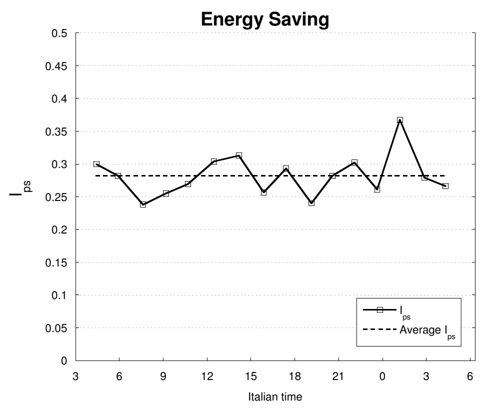


Figure 6.11.: PS-WiFi in the mixed scenario: power-saving performance.

produce a significant effect.

The application-independent policy introduces an average additional delay that is typically in the order of 2-3 sec. (the 90^{th} percentile is less than 10 sec). The increase in the additional delay with respect to the previous scenario is due to the difference between POP3 and SMTP, and HTTP. POP3 and SMTP requires many request/response transactions to exchange an e-mail message. On the other hand, only two transactions are required by the HTTP/1.1 to download a Web page. As explained by the analytical model developed in the following, this has an impact on the additional delay introduced. Though the additional delay is far greater than the delay introduced in the case of Web traffic, it can be nevertheless considered as acceptable for this type of application. The download of e-mail messages takes usually several second to be completed, especially in a mobile environment. Furthermore, e-mail is clearly more delay-tolerant than Web.

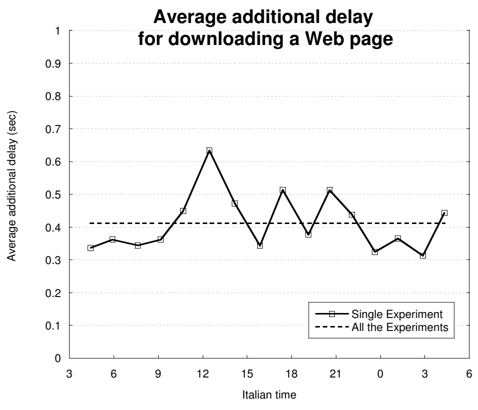
6.3.3. Scenario III: Mixed traffic

From the analysis of the previous scenarios it emerges that the application-independent policy performs well in the presence of a single traffic type, irrespectively of the specific traffic source. Furthermore, in terms of energy saving, it performs better than the local policy, that is an example of application-dependent policy.

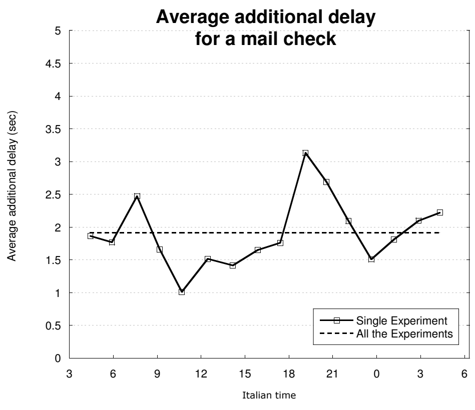
Application-dependent solutions (including the local policy) require a specific software module for each network application. This may be very limiting in a real environment where, typically, many applications may be used, even if not simultaneously. The application-independent policy requires a single software module since it adapts dynamically to the traffic generated by the application(s). For this reason, in the following, we will only consider the application-independent policy.

The objective of this section is to show that this policy exhibits good performance even when there are several concurrent applications. We consider a typical scenario where a user is browsing the Web and, at the same time, periodically (every 5 minutes) connects to the mail server for receiving e-mail messages (if any).

Figure 6.11 shows the I_{ps} index as a function of time in this scenario. Even in a mixed-traffic



(a) Web application



(b) E-mail application

Figure 6.12.: PS-WiFi in the mixed scenario: QoS performance.

scenario, the energy saving with respect to the legacy architecture is significant (on average, 72%). Furthermore, it is very close to the value measured in the Web-only scenario (78%, see Figure 6.8). This can be justified by observing that in the mixed-traffic scenario considered by us the user is continuously browsing the Web, while e-mail traffic can be seen as sporadic with respect to Web traffic. The decrease from 78% to 72% can be justified as follows. When there is a single source of traffic the energy management system quickly adapts to the traffic behavior, thus saving a large amount of energy. In the mixed-traffic scenario the systems adapts to Web traffic in the time interval between two consecutive e-mail checks. When an e-mail check starts the system needs to readapt to the new situation. A similar re-adaptation also takes place after the e-mail transfer has been completed. Such re-adaptations result in a slightly decreased efficiency. The average additional delay introduced is, on average, 0.41 sec for downloading an entire Web page, and 1.9 sec for an e-mail check. With respect to the single-traffic scenarios analyzed above, in the mixedtraffic scenario the average additional delay is almost unchanged for Web traffic and is even lower for e-mail traffic (1.9 sec vs. 2.5 sec). This decrease can be explained as follows. Traffics generated by SMTP and POP3 protocols are characterized by larger (average) inter-arrival times than Web traffic. In the mixed-traffic scenario, estimators are biased to Web traffic and, hence, they tend to underestimate inter-arrival times related to email traffic. This results in a lower additional delay.

6.4. Modeling PS-WiFi behavior

The description of PS-WiFi shows that several parameters affect the system behavior: the throughputs on the wireless and wired networks, the accuracy of the idle-time estimates, the application traffic profile, etc. To better understand their influence on the system performance, in this section we present an analytical model of PS-WiFi. By solving this model we derive closed formulas for I_{ps} and I_{pd}. For the sake of simplicity, the model is derived just in the case of Scenario I, i.e., in

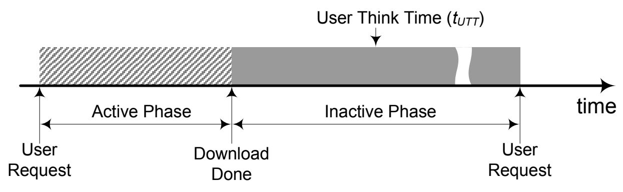


Figure 6.13.: Scheme of the Web traffic as composed by Active and Inactive phases.

Definition	*Symbol*	*Value*	*Unit*
Probability that a Web page contains embedded files	p_{emb}	0.44	-
Average number of embedded files in a Web page	$\overline{N}_{emb}$	1.50	-
Average size of an embedded file	$\overline{D}_{emb}$	6348	bytes
Average size of a main file	$\overline{D}_{mf}$	17496	bytes
Average User-Think-Time length	$\overline{UTT}$	3.25	sec

Table 6.1.: Parameters defining the Web-traffic profile.

presence of Web traffic.

6.4.1. Web-traffic model

Since the system performance is related to the application-level traffic, a preliminary characterization of Web traffic is necessary. A detailed characterization has been provided in Section 5.2. Hereafter we recall the main feature of Web traffic, for the reader convenience.

In this work we exploit the model used in the SURGE simulator [19, 18]. SURGE creates (off-line) a set of Web pages to be stored in a real Web server. Furthermore, it simulates a typical Web user that downloads these Web pages from real (mobile) hosts, and defines the sequence of Web-page downloads in such a way that the traffic generated by the simulated user meets the statistical model of Web traffic presented in [19, 18]. Hereafter, we exploit this model to characterize the traffic generated by a Web user. Figure 6.13 and Table 6.1 describe the traffic model and related parameters derived from SURGE.

Let us focus on a Web-page download. A Web page consists of a main file and zero or more embedded files (e.g., figures). The user requests the Web page to the browser, which downloads all the files from the Web server. When the download is complete, the user reads the contents of the Web page, and then issues another request to the browser. Hence, the Web traffic can be modeled as the sequence of consecutive Web-page downloads. The traffic related to each Web page presents an Active Phase during which the browser downloads the files from the Web server, and an Inactive Phase (or User Think Time, UTT) during which the user reads the contents of the Web page. While the Inactive Phase is completely characterized by the UTT parameter, the Active Phase depends on parameters related to the Web pages. Specifically, the probability that a Web page contains embedded files is hereafter referred to as p_{emb}. Furthermore, the average number of embedded files contained in a Web page[7] is referred to as $\overline{N}_{emb}$. Finally, the average size of a

[7] $\overline{N}_{emb}$ is evaluated by considering only Web pages that contain embedded files.

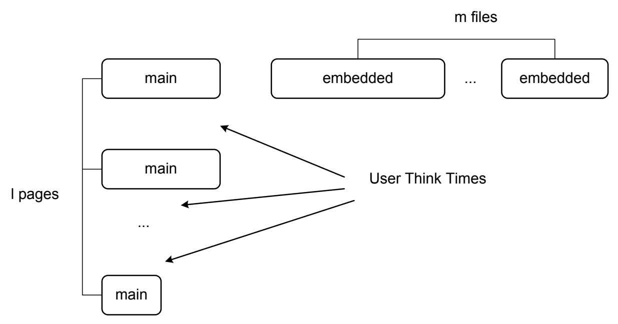


Figure 6.14.: Scheme of the basic block.

main file is $\overline{D}_{mf}$, while the average size of an embedded file is $\overline{D}_{emb}$.

We exploit these parameters (summarized in Table 6.1) to define the reference application-level traffic (a pictorial representation is provided in Figure 6.14). In our model the Web user downloads continuously a set of Web pages, referred to as *basic block*. Between two consecutive downloads, the user waits $\overline{UTT}$ seconds. The pages that compose the basic block are defined to meet the parameters p_{emb}, $\overline{N}_{emb}$, $\overline{D}_{mf}$ and $\overline{D}_{emb}$. Specifically, in our model the first page contains embedded files, while the other pages do not. The average total size of the embedded files must be $\overline{N}_{emb} \cdot \overline{D}_{emb}$, and the number of embedded files must be an integer number. To this end, in our model the first page contains m embedded files, where m is equal to $\lceil \overline{N}_{emb} \rceil$. The size of the first $m-1$ files is $\overline{D}_{emb}$, while the size of the last file is $\overline{D}_{emb} \cdot (m - \overline{N}_{emb})$. Furthermore, as just the first page contains embedded files, the number of Web pages composing the basic block should be $1/p_{emb}$. Moreover, the total size of the basic-block pages must be $\overline{D}_{mf} \cdot (1/p_{emb}) + \overline{D}_{emb} \cdot \overline{N}_{emb}$, i.e., the total size of the main files must be $\overline{D}_{mf} \cdot (1/p_{emb})$. However, since the number of Web pages must be an integer number, the basic block contains l Web pages, where l is equal to $\lceil 1/p_{emb} \rceil$. The main files of the first $l-1$ pages are long $\overline{D}_{mf}$ bytes, while the main file of the last page is long $\overline{D}_{mf} \cdot (l - 1/p_{emb})$ bytes.

In conclusion, in our model a Web user downloads continuously the Web pages contained in the basic block, interleaving each download with a User Think Time. The User Think Time and the basic block definitions guarantee that the application-level traffic has the same average statistics of the traffic generated by the SURGE simulator [19, 18]. Therefore, hereafter we characterize the PS-WiFi behavior by focusing on the download of a single basic block.

6.4.1.1. Idle times characterization

As discussed in Section 6.2.1, PS-WiFi predicts idle-time lengths occurring in the application-level traffic, and manages the wireless interface of the mobile device accordingly. Therefore, at this point, we need to characterize the idle times of the Web traffic we have modeled.

Let us focus on Figure 6.13. Clearly, a first class of idle times is represented by User Think Times. By recalling the definition of long idle time (Section 6.2.1), it is easy to show that User Think Times fall in this category. Furthermore, we need to identify idle times that may occur during

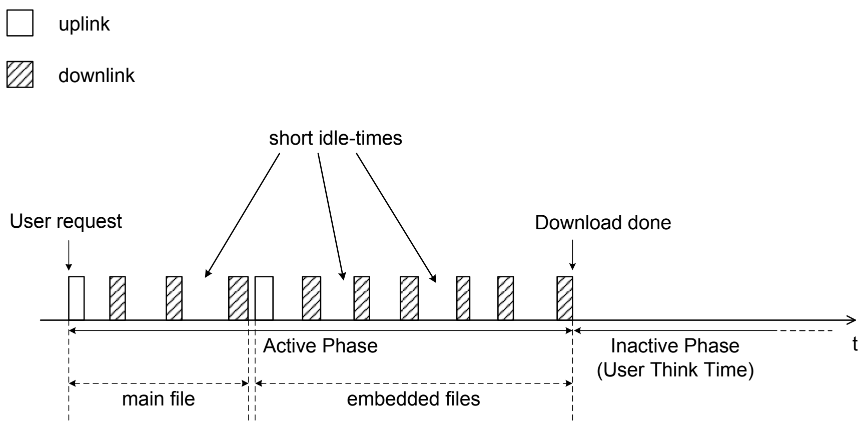


Figure 6.15.: Application-level traffic during the Active Phase

the Active Phases. First of all, it should be noted that during Active Phases the browser and the Web server exchange data without user interventions, and hence idle times must be seen as *short* idle times. By definition, short idle times are related to the behavior of the network protocols, therefore we have to recall which network protocols are used by the Web applications, and how they work in the PS-WiFi architecture (Figure 6.5). Today, Web uses the HTTP/1.1 as the application-level protocol. In our model, we assume that the browser at the mobile device downloads a Web page in two steps (see Figure 6.15): i) the main file is downloaded first, and the list of the embedded files is extracted from it; ii) then, all the embedded files are requested (and downloaded) together[8]. In each step, the shape of the application-level traffic is defined by the underlying protocols, and hence, by the joint effect of PS-PT, STP and TCP. However, as appears from Figure 6.15, the Web traffic is almost mono-directional, in the downlink direction. Therefore, in our model we consider only idle times between packets flowing in the downlink direction, and hence, as discussed in Section 6.2.4, we have to characterize these idle times as observed at the Access Point. Moreover, thanks to the Indirect-TCP architecture, the behavior of the TCP between the Access Point and the fixed host is completely independent of whatever protocol running between the Access Point and the mobile device (i.e., the STP and the PS-PT). Therefore, we conclude that in our model short-idle times are determined only by the behavior of the TCP between the Access Point and the fixed host. To characterize the TCP behavior, it is worth noting that i) HTTP/1.1 uses persistent connections, i.e., the same transport connection can be used to download several files sequentially [34]; ii) Web servers autonomously close persistent TCP connections that remain idle for 15 seconds, and utilize the same TCP connection to serve up to 150 HTTP Requests [11]; and iii) in the model defined by SURGE, User Think Times are less than 15 seconds with very high probability (98%). Therefore, in our model all Web pages composing the basic block are downloaded by means of a single TCP connection. Moreover, we neglect possible slow-start phases, and hence we assume that the TCP connection is in steady-state. Under this hypothesis, the TCP behavior can be modeled as follows [43, 48]: i) once every

[8]This assumption relies on the fact that HTTP/1.1 allows pipelining requests, i.e., consecutive HTTP Requests can be sent back-to-back.

Definition	*Symbol*
Total size of the basic block	B
Number of Web pages in the basic block	l
Average throughput on the wired Internet	$\overline{\gamma}$
Average throughput on the WLAN	$\overline{\gamma}_{wl}$
Energy spent during idle times when using PS-WiFi	C_{it}
Time spent in the idle mode when using PS-WiFi	T_{ON}^{idle}
Transient interval of the wireless interface to switch on	t_{so}
Average number of switching-on events during the basic-block download	S
Average number of switching-on events during a short idle time	S_1
Average number of switching-on events during a long idle time, before the backoff procedure starts	F
Random variable measuring short idle-time lengths	t
Random variable measuring initial estimates of short idle-time lengths	t'
Upper bound of the short idle-time distribution	M
90th percentile of short idle-time lengths	k
Delay added by PS-WiFi to a packet flowing in the downlink direction	d

Table 6.2.: Symbols used in the PS-WiFi model.

RTT the TCP at the Web server sends a fixed number of back-to-back TCP segments[9]; and ii) these back-to-back TCP segments arrive at the Access Point together. Therefore, in our model we assume that short idle times during Active Phases correspond to Round Trip Times between the Access Point and the Web server.

6.4.2. Energy Consumption Modeling

We are now in the position to evaluate the energy spent to download a single basic block, by using either PS-WiFi or the Indirect-TCP architecture without any power management. Hereafter, we assume that the network status is stationary during the download of the basic block (i.e., the throughput and the RTT are stationary). Furthermore, we develop a model of the system behavior in the *average* case, and hence the quantities we derive must be intended as average values. For ease of reading, we summarize in Table 6.2 the parameters used in the model.

The energy spent when using an Indirect-TCP architecture without power management (i.e., C_{I-TCP}) is (proportional to) the total time required to download the basic block. Therefore, the following theorem holds.

Theorem 1 *The energy spent to download a single basic block by using a pure Indirect-TCP approach is*

$$C_{I-TCP} = \frac{B}{\overline{\gamma}} + l \cdot \overline{UTT} = \frac{\overline{D}_{mf} \cdot 1/p_{emb} + \overline{N}_{emb} \cdot \overline{D}_{emb}}{\overline{\gamma}} + l \cdot \overline{UTT}\ , \tag{6.2}$$

where B is the total size (in bytes) of the basic block, and $\overline{\gamma}$ is the average throughput of the (wired) Internet.

Proof. C_{I-TCP} is the time spent to download the basic block, i.e., the time spent downloading the actual data, and the User Think Times. The first term is the ratio between the basic-block size (i.e.,

[9]The number of TCP segment is defined by the average size of the congestion window.

B) and the average throughput experienced by the client. It is worth recalling that in an Indirect-TCP architecture, the throughput achieved at the transport layer is the minimum throughput of the two connections. Thus, if we assume that the fixed Internet is the bottleneck between the server and the client, the average throughput experienced by the client is the throughput of the fixed Internet, i.e., $\overline{\gamma}$. The second term is the sum of the User Think Times occurring within the basic block, i.e., $l \cdot \overline{UTT}$. The final form of C_{I-TCP} follows from the basic-block definition provided in Section 6.4.1. ■

Following the scheme in Figure 6.1, the energy spent using PS-WiFi (i.e., C_{ps}) can be seen as made-up of two components. The first one is related to actual transfers of data composing the basic block. The other one is the energy spent during idle times, throughout referred to as C_{it}. Since idle times are managed as shown in Section 6.2, two factors impact on C_{it}: i) every time the network interface is shut down, t_{so} seconds are paid the next time it is switched on, and ii) when an idle-time estimate (more precisely, an item in the sequence $z^{(i)}$, see Equation 6.1) occurs to be less than t_{so}, the network interface remains in the idle state[10]. Therefore, the following proposition holds.

Proposition 4 *The energy spent by using PS-WiFi is*

$$C_{ps} = \frac{B}{\overline{\gamma}_{wl}} + C_{it} = \frac{B}{\overline{\gamma}_{wl}} + t_{so} \cdot S + T_{ON}^{idle} \ , \tag{6.3}$$

where: i) $\overline{\gamma}_{wl}$ is the average throughput available on the WLAN; ii) S is the average number of times the mobile-device wireless interface is switched on during the basic-block download; and iii) T_{ON}^{idle} is the average time during which the mobile-device wireless interface remains idle when PS-WiFi is adopted.

Proof. Transfers between the Access Point and the mobile device occur at the throughput available on the WLAN. Hence, the energy spent to receive the basic block is $B/\overline{\gamma}_{wl}$. Any other contribution is related to idle times (see Figure 6.1), and is thus comprised in C_{it}. Whenever an idle time occurs, PS-WiFi behaves as shown in Section 6.2. Two kinds of events can happen, i.e., i) the wireless interface is shut down (one or more times), and ii) the wireless interface is let idle, because the (residual) idle time is supposed to be too short. Hence, a first component of C_{it} is defined by the average number of switching-on events occurring during the basic-block download (S), and a second component is the total time during which the wireless interface remains idle (T_{ON}^{idle}). ■

Below we derive a closed form of Equation 6.3, under the assumption of T_{ON}^{idle} being negligible. This hypothesis is true in the experimental testbed used for the validation (Section 6.5). In general, it can be shown that this assumption holds if the average value of the short idle-time estimates is greater than t_{so}. Otherwise, our analysis provides optimistic results, i.e., the upper bound of the energy saved with PS-WiFi. The goal of the following analysis is deriving a closed form for S. To this end, we have now to make some assumptions explicit. Hereafter, we assume that short (long) idle-times are i.i.d. random variables. Moreover, we assume that a short (long)

[10]Since the RTT between the Access Point and the mobile device is typically negligible, and due to the simplicity of the PS-PT protocol [6], the overhead of the PS-PT protocol is assumed to be negligible.

idle-time is independent of any previous long (short) idle-time. Furthermore, we assume that VUA is precise, and hence short idle-time estimates follow the same distribution of real short idle times. Finally, we assume that the sequence of estimates (i.e., $u^{(i)}$, Equation 6.1) related to an idle time is independent of the sequences related to previous idle times. To clarify this assumption, let us focus on Figure 6.7. In that example, the estimate of the idle time between packets A and B is evaluated at time K, and hence PSA evaluates the first item in the sequence $u^{(i)}$ being greater than $K - H$. In other words, that estimate depends on the previous estimate error. In our model we neglect this dependence, and assume that estimates are generated at the point in time when the idle time starts (i.e., H in the example). Since this assumption leads to considering a greater number of switching-on events with respect to the real case, the closed form of S that we derive is an overestimate of the real value. According to these assumptions, we conclude that in our model PS-WiFi regenerates with respect to all points in time when an idle time starts. Hence, to characterize its behavior, we focus on the beginning of an idle time, and analyze the two possible events that may occur: a short idle time or a long idle time. The main result of this analysis is the closed form of C_{ps} provided by Theorem 2. For ease of reading, proofs are postponed to Appendix B.

Theorem 2 *The energy spent to download a single basic block by using PS-WiFi is*

$$C_{ps} = \frac{B}{\overline{\gamma}_{wl}} + t_{so} \cdot \left\{ \frac{B}{\overline{\gamma} \cdot \overline{RTT}} \cdot S_1 + l \cdot \left(F + \lceil \log_2 \overline{UTT} \rceil\right) + p\left(u^{(0)} > t_{so}\right) \right\} , \qquad (6.4)$$

where i) S_1 is the average number of switching-on events occurring during a short idle-time; ii) F is the number of switching-on events occurring during a long idle-time, before the long idle-time estimator is invoked (i.e., after $u^{(2)}$ of Equation 6.1 is generated); and iii) $p\left(u^{(0)} > t_{so}\right)$ is the probability of $u^{(0)}$ being greater than t_{so}.

6.4.3. Modeling the I_{pd} index

The I_{pd} index measures the additional URT introduced by PS-WiFi to download a Web page. To characterize I_{pd}, it is worth focusing on Figure 6.15. Let us introduce the definition of a *transaction* between the Web client and the Web server. A transaction denotes a piece of Web traffic starting with one or more HTTP Requests sent by the client back-to-back, and including all packets sent by the Web server in response to that HTTP Request(s). Hence, the download of a Web page can be modeled as the sequence of two transactions, where i) the main file is downloaded during the first transaction, and ii) all the embedded files are downloaded during the second transaction. Let us now analyze the delay introduced by PS-WiFi to a transaction (see Figure 6.16). When the HTTP Request(s) is generated by the browser, the wireless interface of the mobile device may be shut down. In this case, the HTTP Request(s) is delayed of t_{so} seconds. A further delay can be added at the end of the transaction. As noted in Section 6.4.1.1, the inter-arrival times between packets sent by the Web server depend only on the TCP protocol between the Access Point and the fixed host. In other words, the time instants when these packets arrive at the Access Point are the same whether PS-WiFi is used or not. Therefore, a further delay is added by PS-WiFi if the *last* packet of the transaction is delayed at the Access Point. Hereafter we neglect the possibility that two consecutive packets (e.g., A and B in Figure 6.16) are transferred back-to-back on the

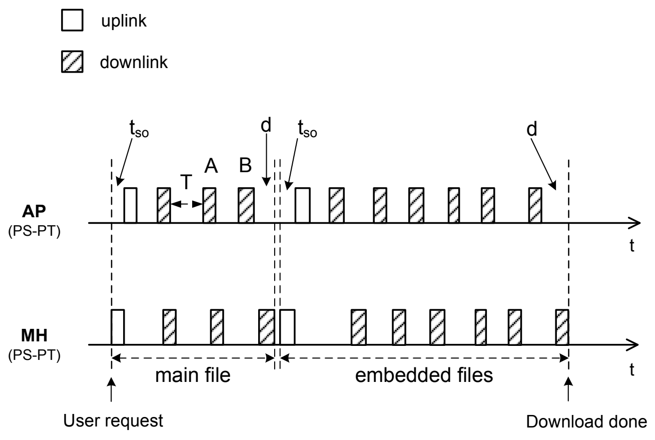


Figure 6.16.: Components of the additional URT.

WLAN, due to overestimation of the previous idle time (e.g., T in Figure 6.16). That is, *all* short idle times are detected and estimated by PS-WiFi. Therefore, a further delay can be introduced to the transaction only as a side effect of the estimate error related to the last short idle time. This delay is hereafter referred to as d.

The analysis of the delay introduced to a transaction allows us to characterize the I_{pd} index. Specifically, from Figure 6.4.1.1 it appears that the embedded files are requested only when the main file is completely downloaded at the mobile device, and hence the second transaction starts only after the last packet of the first transaction arrives at the mobile device. Therefore, I_{pd} can be evaluated as the sum of the delays introduced to each transaction. A closed form for the average value of I_{pd} is provided by the following theorem.

Theorem 3 *The average additional URT introduced by PS-WiFi to download a Web page is*

$$\overline{I_{pd}} = \left(t_{so} + \overline{d}\right) + \left\{t_{so} \cdot p\left(u^{(0)} > t_{so}\right) + \overline{d}\right\} \cdot p_{emb} \ , \tag{6.5}$$

where $\overline{d}$ is the average value of d and p_{emb} is the probability that a Web-page contains embedded files.

Proof. In our model (since User Think Times are assumed to be larger than 1 second) the transaction related to the main file starts when the wireless interface is switched off. Hence, $t_{so} + \overline{d}$ is the average additional delay introduced to the main-file transaction. If the Web-page contains embedded files (i.e., with a probability equal to p_{emb}), the delay related to the embedded-files transaction must be included. A delay equal (on average) to $\overline{d}$ is introduced at the end of the transaction. A further delay equal to t_{so} may be introduced if the transaction starts when the wireless interface is shut down. The probability that this occurs can be derived as follows. When the last packet of the main file is sent to the mobile device, no more data are available to be exchanged on the WLAN. Hence an idle time is detected, and PS-WiFi generates $u^{(0)}$ as its (initial) estimate. However, the browser generates the HTTP Requests for the embedded files immediately, and the second transaction starts. Therefore, the probability that the wireless interface is shut down at this point in time is the probability of $u^{(0)}$ being greater than t_{so}. ■

The evaluation of $\overline{I_{pd}}$ requires the characterization of d. In the following section, we sketch the line of reasoning used to model this quantity, and we provide a closed form for it. For ease of reading, the detailed derivation is presented in Appendix D.

6.4.3.1. Analytical model of d

To evaluate $\overline{I_{pd}}$ we need to characterize d, that is the additional delay PS-WiFi introduces to the last packet of a transaction, due to overestimation of the previous short idle time. As PS-WiFi regenerates with respect to the point in time when an idle time starts, we can analyze d by focusing on the PS-WiFi behavior on a generic short idle time, followed by a packet addressed to the mobile device (i.e., flowing in the downlink direction). To this end, we introduce some assumptions that we exploit hereafter. The estimator of short idle times, is based on the VUA. As noted in Section 6.4.2, VUA allows us to assume that short idle times and their estimates follow the same distribution. In the following, t denotes a random variable measuring short idle times, and t' denotes a random variable measuring the estimates provided by VUA (i.e., t' is equal to $u^{(0)}$ of Equation 6.1). We assume that i) t and t' follow a uniform distribution between 0 and a maximum value, M (i.e., $t \sim t' \in \mathcal{U}\left[0, M\right]$); and ii) t and t' are independent. Moreover, based on the characterization given in Section 6.4.1.1, we instantiate M to two times the average RTT between the mobile device and the Web server, i.e., $M = 2 \cdot \overline{RTT}$. However, to make our analytical approach flexible, hereafter we provide the delay model as a function of M.

The average value of d can be evaluated as the contribution of two components. Specifically, it is the sum of i) the average delay when the initial estimate is too large (i.e., when $t' > t$); and ii) the average delay when the initial estimate is too short (i.e., when $t' < t$). Since t' and t are distributed according to the same law, both $p\left(t' > t\right)$ and $p\left(t' < t\right)$ are equal to $1/2$. These remarks allow to prove the following proposition.

Proposition 5 *The average delay added by PS-WiFi to a packet flowing in the downlink direction can be expressed as*

$$\begin{aligned} \bar{d} = E\left[d\right] = \; & E\left[d\,|t' > t\right] \cdot p\left(t' > t\right) + E\left[d\,|t' < t\right] \cdot p\left(t' < t\right) = \\ = \; & \tfrac{1}{2} \cdot \left(E\left[d\,|t' > t\right] + E\left[d\,|t' < t\right]\right) \end{aligned} \quad . \tag{6.6}$$

The rest of the analysis is devoted to deriving $E\left[d\,|t' > t\right]$ and $E\left[d\,|t' < t\right]$. This is achieved by analyzing the behavior of PS-WiFi in both cases. As this task just requires simple (but quite long) algebraic manipulations, here we only provide the final results, while proofs are postponed to Appendix D.

Lemma 1 *The average delay when t' is greater than t is*

$$E\left[d\,|t' > t\right] = \frac{M^2 - t_{so}^2}{4M} \quad . \tag{6.7}$$

Lemma 2 *The average delay when t' is less than t is*

$$E\left[d\,|t' < t\right] = 0.9 \cdot \frac{k^2 - t_{so}^2}{4M} + 0.1 \cdot \frac{2sec - M - k}{2} \cdot \chi\left(k, t_{so}\right) \quad , \tag{6.8}$$

where $\chi(k, t_{so})$ is an indicator function, defined as

$$\chi(k, t_{so}) = \begin{cases} 1 & \textit{if } 1sec - k > t_{so} \\ 0 & \textit{otherwise} \end{cases} . \tag{6.9}$$

Finally, the above lemmas allow us to provide a closed form for $\bar{d}$, as follows.

Theorem 4 *The average delay added by PS-WiFi to a packet flowing in the downlink direction is*

$$\begin{aligned} \overline{d} &= \frac{1}{2}\left(\frac{M^2 - t_{so}^2}{4M} \cdot u(M, t_{so}) + 0.9 \cdot \frac{k^2 - t_{so}^2}{4M} \cdot u(k, t_{so}) + \right. \\ &+ \left. 0.1 \cdot \frac{2sec - M - k}{2} \cdot \chi(k, t_{so})\right) \end{aligned} , \tag{6.10}$$

where $u(x, y)$ is the classical step function:

$$u(x, y) = \begin{cases} 1 & \textit{if } x \geq y \\ 0 & \textit{otherwise} \end{cases} . \tag{6.11}$$

6.5. Model validation

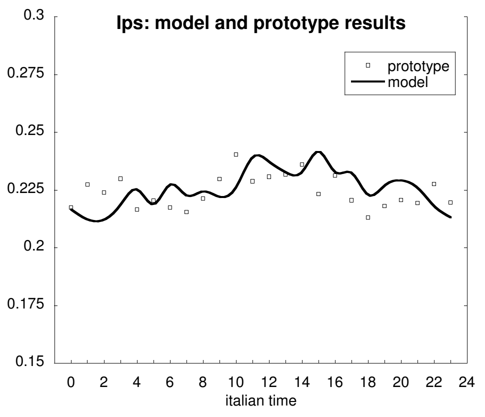
To validate the analytical model derived in Section 6.4.1 we compare its predictions with measurements taken from a real Internet prototype (details about the prototype can be found in [6]). Specifically, we compare the I_{ps} and $\overline{I_{pd}}$ measurements presented in [6] with the predictions provided by our model. To avoid fluctuations in the experimental results, we averaged measurements on one-hour windows (details about the methodology used to aggregate measurements can be found in [7]). It is worth noting that, due to the experiment setup, the only parameter that changes significantly among different hours is the throughput of the wired Internet, i.e., $\overline{\gamma}$. Indeed, i) in each experiment the application-level traffic meets the statistics shown in Table 6.1; ii) $\overline{\gamma}_{wl}$ and $\overline{RTT}$ show very little fluctuations (and, thus, also M and k can be assumed to be constant); and iii) due to the assumptions about the distribution of t and t', the parameters S_1, F and $p\left(u^{(0)} > t_{so}\right)$ in Equation 6.4 can be approximated as constant terms. Table 6.3 summarizes the values of the parameters that we use to validate our analytical model.

Figure 6.17(a) shows the hourly average values of I_{ps} and $\overline{I_{pd}}$ measured by using the prototype. We also plot the I_{ps} and $\overline{I_{pd}}$ figures derived from the analytical model. As far as the I_{ps} index (left-side plot), the model and the prototype provide very close results: the difference is always less than 9% of the prototype results. Furthermore, we have compared the daily average values of I_{ps} with the predictions of the model. Specifically, we have set $\overline{\gamma}$ in Equation 6.4 to the average daily throughput experienced by the prototype. The results obtained (not reported here) show that the difference between the model results and the prototype measurements is less than 1%.

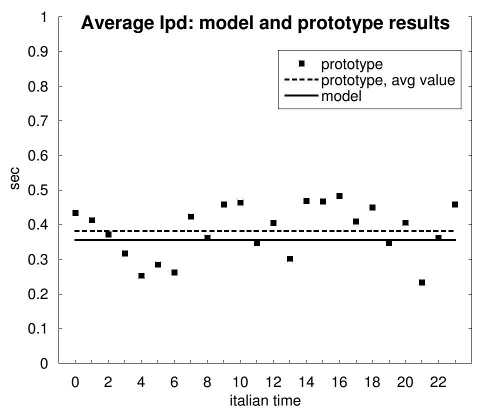
As far as the $\overline{I_{pd}}$ index (Figure 6.17(b)), the results show that prototype values vary during the day, i.e., the prototype is sensitive to variations of $\overline{\gamma}$. Indeed, $\overline{\gamma}$ is sensitive to two parameters, i.e., i) congestions in the Internet, that reduce the TCP window size; and ii) variations of the RTT between client and server. As discussed in Section 6.4.3.1, the additional URT is affected by RTT variations. However, we have no sufficient information to include the precise RTT pattern in

Definition	*Symbol*	*Value*	*Unit*
Total size of the basic block	B	49264	bytes
Number of Web pages in the basic block	l	3	-
Average throughput over the wireless link	$\overline{\gamma}_{wl}$	11	Mbps
Average number of switching-on events in a short idle time	S_1	1.55	-
Average number of switching-on events in a long idle time before the backoff procedure starts	F	3	-
Probability that the initial estimate of a short idle time is greater than t_{so}	$p\left(u^{(0)} > t_{so}\right)$	1	-
Network RTT between the client and the Web server	$\overline{RTT}$	0.3	sec
Upper bound of t and t' distributions	M	0.6	sec
Switching-on transient interval of the wireless interface	t_{so}	0.1	sec
90th percentile of t and t'	k	0.54	sec
Indicator function of k and t_{so} relative values	$\chi\left(k,\, t_{so}\right)$	1	-

Table 6.3.: Parameters used to validate the analytical model.



(a) I_{ps}



(b) I_{pd}

Figure 6.17.: Hourly average I_{ps} and $\overline{I_{pd}}$ obtained from the model and the prototype, respectively.

the analytical model. As a consequence, the $\overline{I_{pd}}$ model allows us to measure the additional URT related to the average RTT. Specifically, we compare the model prediction with the daily average value of $\overline{I_{pd}}$ (see the dashed line in the plot). The difference is about 7% of the prototype result.

6.6. Sensitiveness Analysis

The above results show the accuracy of our analytical model. Hereafter, we use this model to investigate the sensitiveness of PS-WiFi to two Internet key parameters, i.e. the (wired) Internet throughput (i.e., $\overline{\gamma}$) and the RTT. As noted in the previous section, the throughput depends on both the network RTT and the TCP window size. Both these parameters affect the I_{ps} index, since it depends on $\overline{\gamma}$ (see Equations 6.2 and 6.4). On the other hand, the $\overline{I_{pd}}$ index is only affected by RTT variations, as shown by Equations 6.5 and 6.10. Therefore, below we analyze I_{ps} as a function of $\overline{\gamma}$, and $\overline{I_{pd}}$ as a function of RTT.

6.6.1. Power-Saving Sensitiveness

Based on Equations 6.2 and 6.4 we derive the I_{ps} index as a function of $\overline{\gamma}$. Specifically, after simple manipulations, $I_{ps}\left(\overline{\gamma}\right)$ becomes:

$$I_{ps}\left(\overline{\gamma}\right) = \frac{a\overline{\gamma} + b}{c\overline{\gamma} + d} \quad , \tag{6.12}$$

where a, b, c and d group terms that we have assumed to be constant (see Section 6.5):

$$\begin{cases} a \triangleq \frac{B}{\overline{\gamma_{wl}}} + t_{so} \cdot \left\{ l \cdot \left(F + \left\lceil \log_2 \overline{UTT} \right\rceil\right) + p\left(u^{(0)} > t_{so}\right)\right\} \\ b \triangleq \frac{B}{\overline{RTT}} \cdot S_1 \cdot t_{so} \\ c \triangleq l \cdot \overline{UTT} \\ d \triangleq B \end{cases} \quad . \tag{6.13}$$

Figure 6.18(a) shows I_{ps} as a function of $\overline{\gamma}$. It clearly appears that when $\overline{\gamma}$ increases, I_{ps} decreases, and this means that the PS-WiFi saves more energy. This result is somehow counter-intuitive, since one expects that the best power-saving is achieved when $\overline{\gamma}$ is low. In this case the overall idle time during the basic block download is at its maximum value. However, the I_{ps} behavior in the above plot can be explained as follows. As shown in Equations 6.2 and 6.4, variations of $\overline{\gamma}$ affect both C_{I-TCP} and C_{ps}. In the Indirect-TCP architecture, when $\overline{\gamma}$ increases, the time needed to fetch the basic block from the Web server (i.e., $B/\overline{\gamma}$) decreases, and C_{I-TCP} decreases accordingly. On the other hand, the dependence of C_{ps} on $\overline{\gamma}$ is as follows. As highlighted in Section 6.5, $\overline{\gamma}$ is strictly related to the the TCP window size. Since RTT is almost stable, large TCP windows mean high $\overline{\gamma}$ values, while narrow TCP windows correspond to low $\overline{\gamma}$ values. Furthermore, if the TCP window size increases, the number of $RTTs$ needed to fetch the basic block drops, since more bytes are downloaded in a single RTT. Thus, the number of switching-on events during the basic-block download decreases, and hence, we can conclude that the more $\overline{\gamma}$ increases, the more C_{ps} decreases. Since both C_{I-TCP} and C_{ps} benefit from increases of $\overline{\gamma}$, the I_{ps} pattern is defined by the parameters a, b, c and d. Specifically, in the Internet configuration of our experiments, when $\overline{\gamma}$ increases, C_{ps} decreases more than C_{I-TCP} does, and hence I_{ps} drops.

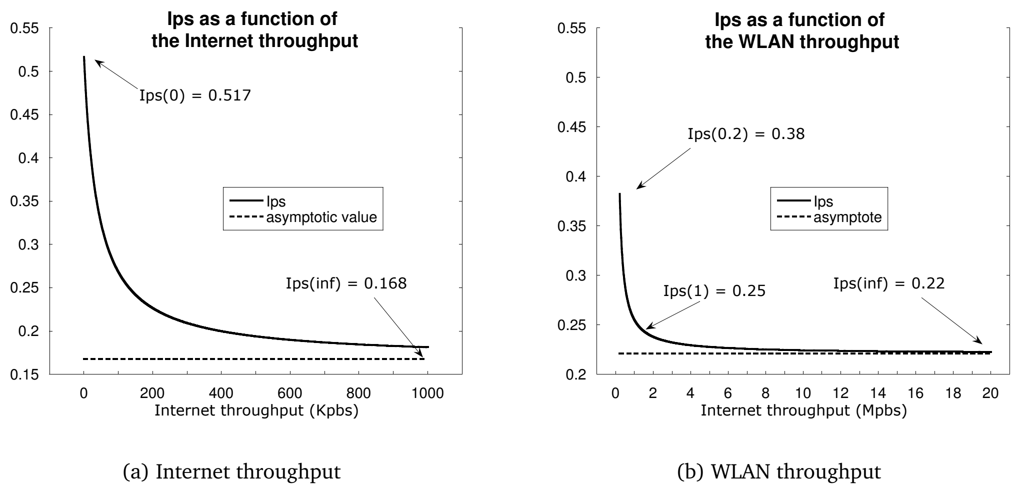


Figure 6.18.: I_{ps} as a function of the throughput on the Internet (a) and on the wireless LAN (b), respectively.

It is worth noting that Figure 6.18 highlights the theoretical lower and upper bounds of I_{ps} (for varying $\overline{\gamma}$ values). In the Internet configuration that we experienced, I_{ps} ranges between 0.517 (when $\overline{\gamma} \to 0$) and 0.168 (when $\overline{\gamma} \to \infty$). Therefore, with respect to the Indirect-TCP architecture, our power-saving system guarantees energy savings that are always above 48%, and raise up to 83%. However, if we focus on realistic throughput values (i.e., between 50 Kbps and 1 Mbps), energy saving does not vary sharply, since it ranges between 68% and 82%.

Furthermore, Figure 6.18(b) shows the dependence of I_{ps} on the throughput available on the wireless LAN. Also in this case, the higher the throughput, the higher the power saving achieved. An interesting feature of this plot is that the I_{ps} curve flattens for relative small values of the throughput (i.e., around 1 Mbps). This can be explained as follows. The energy consumption of PS-WiFi depends on two factors, i.e., the time spent to transfer the data on the wirless LAN, and the time spent in switching on events. As the throughput on the wireless LAN increases, the former component decreases, and eventually becomes negligible with respect to the latter one. In a nutshell, for high throughput, the energy consumption of PS-WiFi mostly depends on the number of switching on events, and – ultimately – on the idle-time predictions accuracy (at least, for the amount of traffic downloaded in our scenario).

6.6.2. QoS Sensitiveness

In this section we analyze the dependence of $\overline{I_{pd}}$ on the average Round Trip Time, i.e., $\overline{RTT}$. As a preliminary step, it is necessary to define the range of valid $\overline{RTT}$ values. Specifically, PS-WiFi defines 1 sec as the upper bound of short idle times. Therefore, 1 sec is also the upper bound of both the short idle time and estimate distributions (i.e., $M \leq 1$ sec). Since in our model M is equal to $2 \cdot \overline{RTT}$, $\overline{RTT}$ must be less than 0.5 sec. On the other hand, we can use 0 as the lower bound – or, more precisely, as the theoretical lower limit – of $\overline{RTT}$.

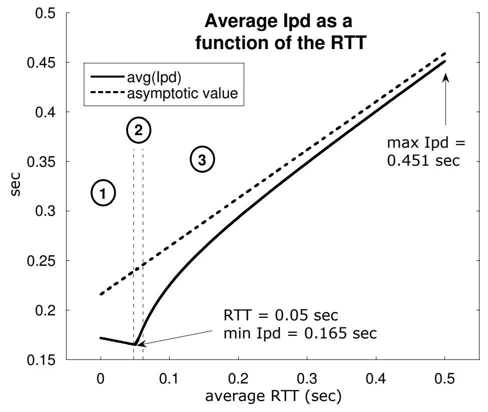


Figure 6.19.: $\overline{I_{pd}}$ as a function of the $\overline{RTT}$ value.

It must be also pointed out that t_{so} is the lower bound for $\overline{I_{pd}}$. Specifically, even if the estimator error were always equal to 0, the mobile device would switch the wireless interface on at least once every Web-page download (i.e., when the user sends a new Web page request). Therefore t_{so} represents an additional URT that can never be eliminated when using PS-WiFi. Finally, for the sake of simplicity, hereafter we assume that t_{so} =0.1 sec holds. Therefore, since t_{so} is equal to 0.1 sec and $\overline{RTT}$ is less than 0.5 sec, $\chi\left(k,\, t_{so}\right)$ is always equal to 1. However, it is easy to extend the analysis also to general t_{so} values.

From Equations 6.5 and 6.10 we derive the plot shown in Figure 6.19. In Figure 6.19 we can observe three regions, i.e., i) the part where $M = 2 \cdot \overline{RTT}$ is less than t_{so}, ii) the part when $M = 2 \cdot \overline{RTT}$ is between t_{so} and $t_{so}/0.9$, and, finally, ii) the part where $M = 2 \cdot \overline{RTT}$ is greater than $t_{so}/0.9$. Indeed, the second region is very small, and can hardly be distinguished in Figure 6.19.

In the first region, both $u\left(M,\, t_{so}\right)$ and $u\left(k,\, t_{so}\right)$ are equal to 0. Furthermore, $\overline{I_{pd}}$ is a decreasing function of $\overline{RTT}$, and reaches its minimum value ($\min \overline{I_{pd}}$ =0.165 sec when $\overline{RTT}$ =0.05 sec). This behavior can be explained by recalling the idle-time estimator algorithm. In this region, an additional delay is added to a packet flowing in the downlink direction only when the idle time is greater than k, and hence PS-WiFi provides $u^{(2)} = 1\,sec$ as the updated estimate. Otherwise, since both $z^{(0)} = t'$ and $z^{(1)} = k - t'$ are less than t_{so}, no delay is introduced. Since M is less than t_{so}, $p\left(t' > t_{so}\right)$ is equal to 0, and hence $\overline{I_{pd}}$ becomes equal to $\overline{d} \cdot \left(1 + p_{emb}\right) + t_{so}$. Moreover, $\overline{d}$ is basically defined by Equation D.12 (see Appendix D), which is a decreasing function of $\overline{RTT}$. Specifically, $\overline{d}$ is equal to $1\,sec - \left(M + k\right)/2$, and it is at its lowest value when $\overline{RTT}$ (and thus M) is at its maximum value.

In the second region, $u\left(M, t_{so}\right)$ is equal to 1, while $u\left(k,\, t_{so}\right)$ is equal to 0. Since M is greater than t_{so}, there are two differences with respect to the previous region, i.e. i) $p\left(t' > t_{so}\right)$ is equal to $\left(M - t_{so}\right)/M$, and ii) $\overline{d}$ becomes an increasing function of $\overline{RTT}$. Therefore, the following equation holds:

$$\overline{I_{pd}} = t_{so} + \overline{d} + \left\{ t_{so} \cdot \frac{M - t_{so}}{M} + \overline{d} \right\} \cdot p_{emb}\ . \tag{6.14}$$

Therefore it is easy to show that $\overline{I_{pd}}$ increases with $\overline{RTT}$, as $\overline{d}$ is an increasing function of $\overline{RTT}$

(see Appendix D).

Finally, in the third region, both $u\,(M,\,t_{so})$ and $u\,(k,\,t_{so})$ are equal to 1. In this case, $\overline{I_{pd}}$ can again be evaluated by means of Equation 6.14. However, when $\overline{RTT}$ increases, $\overline{d}$ increases more quickly than in the second region, since $u\,(k, t_{so})$ is equal to 1 (see Equation 6.10). Therefore, also $\overline{I_{pd}}$ increases more quickly than in the second region. Moreover, in this region it reaches its maximum value ($\max \overline{I_{pd}} = 0.451\,sec$, achieved when $\overline{RTT} = 0.5\,sec$).

As a final remark, it is worth noting that the $\overline{I_{pd}}$ figure in the third region can be well approximated by a linear increasing function, that grows as $0.487 \cdot \overline{RTT}$. To summarize, we can conclude that increases of $\overline{RTT}$ have a moderate impact on the additional URT.

6.7. Summary

We have presented a power-saving system (PS-WiFi) based on an application-independent approach. Specifically, PS-WiFi does not rely on a-priory assumptions about the application(s) behavior, but dynamically adapts its power-saving policy to the ongoing traffic pattern. We have tested PS-WiFi in a real-Internet prototype, by running either Web and e-mail applications on top of it. We have also tested PS-WiFi in the case of concurrent applications. We have comparted its power-saving performance with respect to PS-Web (an application-dependent power-saving system). Clearly PS-Web performs better, but the performance degradation of PS-WiFi is reasonable. To assess PS-WiFi performance more completely, we have developed an analytical model of its behavior. The analytical model is used to analyze the performance of PS-WiFi assuming mobile Web access applications. We have compared our solution with an Indirect-TCP architecture without power management by evaluating two performance indexes, i.e., the energy spent in downloading a Web page and the related transfer-time. We have derived closed formulas that allowed us to evaluate both performance indexes. We have validated the model by comparing its predictions with the results obtained from the prototype. Experimental results have shown that our model is highly accurate, since the average difference with respect to experimental measurements is about 1% for the energy-saving index, and about 7% for the transfer-time index. Therefore, our model is a valid tool to characterize the behavior of PS-WiFi with respect to key parameters, such as the application traffic profile, the throughput on the wireless and wired networks, the RTT between the client and the server, etc. Specifically, we have presented a sensitiveness analysis with respect to two Internet key parameters, i.e., the throughput on the wired network and the RTT. This analysis has shown that energy saving varies from 48% up to 83%, when the throughput increases from 0 to ∞. However, when focusing on more realistic throughput ranges (i.e., between 50 Kbps and 1 Mbps), the energy saving does not vary sharply, and is always greater than 68%. Finally, the average additional transfer-time is a slightly increasing function of the average RTT. However, we can conclude that PS-WiFi never affects the QoS perceived by Web users, since the average additional transfer-time is always less than 0.5 sec.

To conclude this part of the work, we can claim that PS-Web is a better tool in dedicated environments, when the set of applications can be known at design time. On the other hand, PS-WiFi is the best candidate in generic environments, since it works irrespective on the application(s) used by the mobile user(s).

Part III.

A Cross-layer Approach to Power Management

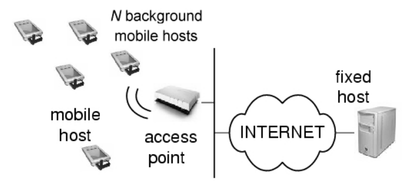
7. From 802.11 PSM to Cross-Layer Power Management

7.1. Overview

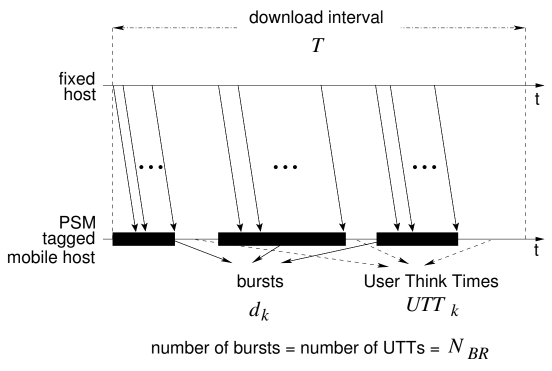
So far, we have used middleware-level strategies for power management. The advantage of such an approach is the independence on the specific wireless technology used in the WLAN. In addition, it provides a good solution for heterogeneous wireless environments, as well. The main drawback is that, since no specific wireless technology is assumed, low-power states of the network interface – which are typically technology dependent – cannot be exploited. In other words, the power-saving policies we have proposed can just switch the wireless interface completely off to conserve energy. As highlighted before, the cost of each switch-on event is around 100 ms. This means that, for idle times less than 100 ms, the wireless interface remains active. Clearly, this may lead to subotimal power-saving, since 100 ms is a reasonable value for interarrival times.

To avoid this drawback, different low-power state of the wireless interface should be used. For example, the IEEE 802.11 standard [35] defines a *sleep* operating mode where just a portion of the card circuitry is powered, while most of the card is off. Reverting the card to full active mode is much more quick than activating the card from the off mode – typically it costs just 1 ms. Hence, such operating mode is more suited for short idle times. On the other hand, the sleep mode still consume some energy, and hence it may not be the best operating mode for long idle times.

Based on these remarks, we now focus on the de-facto standard technology for Wi-Fi hotspots, i.e., 802.11 ("Wi-Fi"), and evaluate the Power-Saving Mode, which is defined within the standard to conserve mobile hosts' energy. We provide an extensive characterization of PSM performance, highlighting strengths, weaknesses, and possible improvements. We consider the typical Wi-Fi hotspot scenario depicted in Figure 7.1(a) and 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 observation that the traffic generated by these applications – Web most of all – represents the lion's share of the today Internet traffic, and Web is very likely to be the largely dominant application also in the near-future Internet [15]. As noted in the previous parts of this work, the traffic generated by the reference applications is bursty, and presents different types of idle times (Figure 7.1(b) shows a snapshot of such traffic). Specifically, idle times inside bursts (referred to as *interarrival times*) are typically very short, less than 1 s. On the other hand, bursts are spaced by longer idle times (referred to as *User Think Times*), that can last for 60 s and beyond. As the goal of PSM is reducing the power consumption during idle times, we extensively analyze its behavior with respect to the two aforementioned classes.



(a) Wi-Fi hotspot scenario



(b) Application-level traffic

Figure 7.1.: The reference environment.

We find that 802.11 PSM is very effective with respect to interarrival times, by producing power savings in the order of 80-90%. The amount and length of idle times inside bursts 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 application and network parameters. Specifically, we find that the power saving PSM achieves does not significantly depend on the size of bursts. With respect to the networking environment, we consider the two main parameters that define the transport-level throughput, i.e., the segment-loss probability (p_l^{tcp}), and the Round Trip Time (RTT) between the TCP-connection endpoints. Furthermore, we also analyze the power-saving performance of PSM in presence of variable number of users within the hotspot, i.e., in presence of variable MAC-level congestion. As expected, the power consumption of PSM increases when either the TCP segment-loss probability, or the Round Trip Time, or the MAC-level congestion increases. However, the performance degradation when PSM is active is far lower than when it is not. Finally, by means of analytical results, we show that, during burst downloads, the power consumption achieved by PSM closely approximates the power consumption achieved by the *ideal* policy, i.e., a power-saving policy that knows in advance the length of interarrival times. Specifically, the difference between the PSM and the ideal power manager is never greater than 20% of the ideal power consumption.

On the other hand, we find that PSM is not a good approach to maximize power saving during User Think Times. The main reason is that PSM just exploits the sleep state of the wireless interface, and periodically wakes up the wireless interface to poll the Access Point for new data. For such long idle times, using the off state (i.e., switching the wireless interface off) is more energy efficient. We propose a cross-layer extension to the standard PSM that exploits this observation. In particular, our *Cross-layer Power Manager* (CPM) is able to detect the beginning of both User Think Times and bursts. During bursts it activates the 802.11 PSM, while during User Think Times it switches the wireless interface off. The Cross-layer Power Manager integrates PSM and middleware-layer policies such as those presented in the previous parts of this work. As such, it exploit MAC-level information and middleware/application level information to conserve energy, and integrates power-saving policies implemented at both levels. Therefore, it uses a cross-layer approach to power saving. The additional power saving achieved by CPM with respect to PSM ranges from 20% to 90%, depending on the User Think Time length, and the bursts' size. Moreover, the Cross-layer Power Manager does not require hardware modifications at either the wireless interface or the Access Point, making it suitable to be simply implemented in current commercial devices.

7.2. Networking and Evaluation Environment

Before proceeding with the analysis, it is worth focusing again on the reference environment used in our work, to highlight some assumptions used to model the 802.11 PSM.

At the application level, our scenario is as follows (see Figure 7.1(b)). A mobile, PSM-enabled, mobile host[1]in a Wi-Fi hotspot downloads a predefined number of *bursts* (say, 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 User Think Times no traffic flows between the server and the client. Though very simple, this model of application-level traffic captures the typical behavior of non-interactive applications' user. For example, Web users download a page (i.e., a burst) and then read the page contents without generating any traffic on the network.

The mobile host and the fixed server communicate through a standard TCP/IP stack. We assume TCP-Reno, without delayed acks [53]. The whole download is supported by a unique TCP-connection, i.e., the same connection persists among different burst downloads. Assuming persistent connections is aligned with the ever more diffused strategies aimed at boosting Internet servers' performance (see, for example, HTTP/1.1 [34, 27]). On the other hand, assuming that a single TCP connection is used allows us to keep the analysis of both PSM and CPM simple. Extending the analysis to the case of concurrent TCP connections is easy. Specifically, in the rest of the paper we highlight how CPM can be extended in that case. It is well-known that the performance of the legacy TCP/IP architecture, both in terms of throughput and in terms of energy consumption, can be very scarce in WLAN environments (see the survey in [17]). However, TCP/IP is currently the only off-the-shelf solution for Wi-Fi hotspots.

In our scenario, the hotspot is populated by other N (background) mobile hosts (Figure 7.1(a)). 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. We also assume that no hidden node phenomena occur, i.e., all mobile hosts can hear transmissions of each other. As discussed in Section 7.5.2.3, by varying the number or active mobile hosts (i.e., M) we can analyze the sensitiveness of PSM to the congestion level in the hotspot, and – therefore – its scalability with respect to the number of users sharing the same Access Point.

The scenario described so far is the reference environment of our analysis. Furthermore, the analysis is carried out by means of the following performance figures:

- ✧ E_{NO_PSM}: the average energy spent to download N_{BR} bursts from the fixed to the mobile host, when PSM is not active (i.e., when no power-management is used);

[1]Throughout the paper, this mobile host is referred to as the (PSM) tagged mobile host

- E_{PSM}: the average energy spent to download N_{BR} bursts from the fixed to the mobile host, when PSM is active;
- $R\left(E_{PSM}, E_{NO\_PSM}\right)$: the ratio between the above indexes, i.e., the fraction of energy spent when PSM is active, with respect to the case when no power management is used.

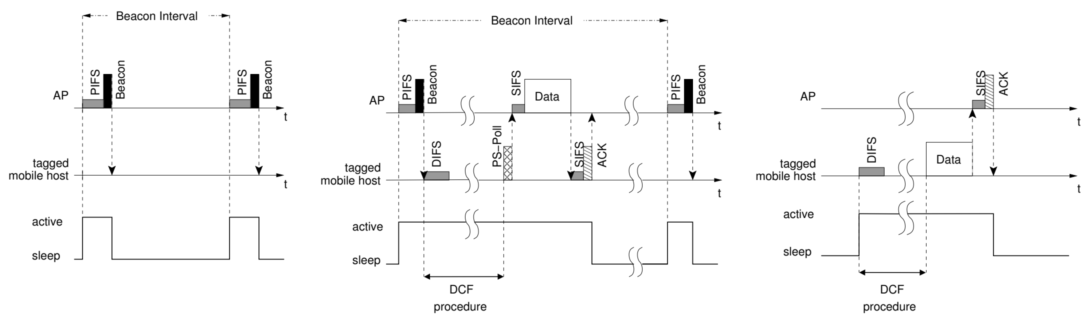
Throughout the paper, we analyze energy consumption breakdowns for the different power managers under investigation. In that cases, more specific performance indexes are defined. When meaningful, the index $R\left(\cdot,\cdot\right)$ is also applied to couples of those indexes.

Our analysis relies on both analytical and simulation results. Specifically, we derive an analytical model that provides closed form expressions of E_{PSM} and E_{NO_PSM}. To assess the model accuracy, we compare its predictions against results from a simulation model. The simulator, which extends the model used in [22], implements the reference environment described in this section. Specifically, we it simulates a full-compliant 802.11 hotspot, populated with a variable number of background mobile hosts. Moreover, it also simulated a full-compliant TCP-Reno between the mobile and fixed hosts. Internet Round Trip Times (RTT) are sampled from an exponential distribution, and, to simulate packet losses at the Internet routers, TCP segments are randomly dropped with probability p_l^{tcp}. To consider significant values for the burst sizes and the User Think Time lengths, we focus on the Web traffic. In particular, we consider the statistical models of the Web traffic presented in the well-known works of Crovella et al. [19, 25]. Throughout the paper, simulation results are presented together with the corresponding values of the simulator parameters. To increase results reliability, each simulation experiment has been replicated 10 times, and confidence intervals for the performance figures have been computed (95% confidence level). Since, as shown in Section 7.4, results from the analytical and the simulation models are in agreement, in the following we exploit both of these tools to characterize the PSM behavior.

Before proceeding with the analysis, the next section provides an overview of the power-saving algorithm defined by the standard IEEE 802.11 [35].

7.3. 802.11 Power-Saving Mode (PSM)

The IEEE 802.11 standard [35] defines two possible operating modes for the wireless interface, i.e. the *active* mode and the *sleep* mode. While in active mode, the wireless interface is able to exchange data, and can be in the *receiving*, in the *transmit*, or in the *idle* state (i.e., it simply overhears the traffic on the channel). Due to the 802.11 MAC protocol, the energy consumption of the active mode depends very little on the operating state. The well-known measurements in [29] prove this claim. Hence, the power consumption of the active mode is typically approximated with a constant value (e.g., 750 mW for Enterasys Networks RoamAbout interfaces [38]). Throughout the paper we use the same approximation. On the other hand, while in sleep mode, only few components of the wireless interface are supplied by the battery (e.g., the clock that maintains synchronization with the Access Point). Therefore, the wireless interface is not able to exchange data, but its power consumption is at least one order of magnitude lower than in the active mode (e.g., 50 mW, [38]).



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

Figure 7.2.: PSM operations.

The objective of the 802.11 PSM is to let (the wireless interface of) a mobile host[2] in the active mode only for the time necessary to exchange data, and to 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 informs the Access Point on 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 7.2(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 7.2(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 7.2(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 the basis for the analytical characterization of PSM that have been derived in [4]. For the reader convenience, in the following section we report the main results of this analysis.

[2]In this paper, the focus is on the wireless interface management. For ease of reading, in the following the operating mode of the mobile host means the operating mode of the *wireless interface* of the mobile host.

7.4. Analytical model of 802.11 PSM

According to the reference scenario, 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 server. To this end, we use the following modeling approach. We replicate n times the download of N_{BR} bursts, and we focus on the generic i-th replica, where i belongs to $\{1, \ldots, n\}$. In the following, $E^{(i)}_{PSM}$ and $E^{(i)}_{NO\_PSM}$ denote the energy spent during the i-th replica when PSM is active or not, respectively. We derive closed form expressions of $E^{(i)}_{PSM}$ and $E^{(i)}_{NO\_PSM}$, and we show that the sets $\left\{E^{(i)}_{PSM}\right\}_i$ and $\left\{E^{(i)}_{NO\_PSM}\right\}_i$ are composed by i.d. random variables. Therefore, we can express E_{PSM} and E_{NO_PSM} as follows:

$$\begin{cases} E_{PSM} & = \lim\limits_{n\to\infty} \dfrac{\sum_{i=1}^{n} E^{(i)}_{PSM}}{n} \\ E_{NO\_PSM} & = \lim\limits_{n\to\infty} \dfrac{\sum_{i=1}^{n} E^{(i)}_{NO\_PSM}}{n} \end{cases} \tag{7.1}$$

By substituting in Equation 7.1 the expressions of $E^{(i)}_{PSM}$ and $E^{(i)}_{NO\_PSM}$ we finally derive the closed form expressions of E_{PSM} and E_{NO_PSM}. The detailed derivation is provided in the following section. Then, Section 7.4.2 is devoted to validate the analytical results. For the reader convenience, it is worth firstly clarifying the notation used hereafter. In detail, Table 7.1 defines all the relevant symbols.

7.4.1. Modeling E_{PSM} and E_{NO_PSM}

A simple expression of $E^{(i)}_{PSM}$ and $E^{(i)}_{NO\_PSM}$ is provided in Propositions 6 and 7.

Proposition 6 *The energy spent by the tagged mobile host during the i-th replica when PSM is not activated is*

$$E^{(i)}_{NO\_PSM} = T^{(i)} \cdot P_{ac} \;, \tag{7.2}$$

where $T^{(i)}$ is the length in time of the i-th replica (throughout referred to also as the download interval), and P_{ac} is the power consumed by the tagged mobile host in the active mode.

Proof. When the PSM is not used, the tagged mobile host is always in the active state. Equation 7.2 derives immediately from the assumption that the power consumption in the active state is always the same, either the mobile host is transmitting, or receiving, or idle. ■

Proposition 7 *The energy spent by the tagged mobile host during the i-th download interval when PSM is activated is*

$$E^{(i)}_{PSM} = T^{(i)}_{ac} \cdot P_{ac} + T^{(i)}_{sl} \cdot P_{sl} = T^{(i)}_{ac} \cdot (P_{ac} - P_{sl}) + T^{(i)} \cdot P_{sl} \;, \tag{7.3}$$

where: i) $T^{(i)}_{ac}$ and $T^{(i)}_{sl}$ are the time intervals in which the tagged mobile host is respectively active and sleeping, during the i-th replica; and ii) P_{sl} is the power consumed by the tagged mobile host in the sleep mode.

Symbol	*Explanation*
$T^{(i)}$	length in time of the i-th replica (also referred to as download interval)
$T_{ac}^{(i)}$	total time during which the tagged mobile host is active during the i-th replica (including transition times from the sleep mode)
$T_{sl}^{(i)}$	total time during which the tagged mobile host is sleeping during the i-th replica
$d_k^{(i)}$	size of the k-th burst downloaded in the i-th replica ($k = 1, \ldots, N_{BR}^{(i)}$)
$td_k^{(i)}$	time required by the tagged mobile host to dowload the
	k-th burst in the i-th replica, when PSM is not active
$UTT_k^{(i)}$	length in time of the k-th User Think Time in the i-th replica ($k = 1, \ldots, N_{BR}^{(i)}$)
$N_{seg}^{(i)}$	number of TCP data segments downloaded by the tagged mobile host in the i-th replica
$N_{ack}^{(i)}$	number of TCP ack segments downloaded by the tagged mobile host in the i-th replica
$ts_j^{(i)}$	time required by the tagged mobile host to download the j-th TCP data segment during the i-th replica ($j = 1, \ldots, N_{seg}^{(i)}$)
$ta_r^{(i)}$	time required by the tagged mobile host to download the r-th TCP ack segment during the i-th replica ($r = 1, \ldots, N_{ack}^{(i)}$)
BI	length in time of a Beacon interval
$N_b^{(i)}$	number of Beacon intervals during the i-th replica
$tb_l^{(i)}$	time required to send the l-th Beacon frame during the i-the replica ($l = 1, \ldots, N_b^{(i)}$)
t_{sa}	time required by the tagged mobile host to switch from sleep to active
P_{ac}	power drained by the tagged mobile host in the active mode
P_{sl}	power drained by the tagged mobile host in the sleep mode

Table 7.1.: Symbols used in the model

Proof. As explained in Section 7.3, a tagged mobile host activating the PSM operates either in the active or in the sleeping mode. During the i-th replica, the energy spent in the active mode is $T_{ac}^{(i)} \cdot P_{ac}$, while the energy spent in the sleeping mode is $T_{sl}^{(i)} \cdot P_{sl}$. By definition, the sum of $T_{ac}^{(i)}$ and $T_{sl}^{(i)}$ is the duration of the i-th replica (i.e., $T^{(i)} = T_{ac}^{(i)} + T_{sl}^{(i)}$). The final form of Equation 7.3 is derived by expressing $T_{sl}^{(i)}$ as $T^{(i)} - T_{ac}^{(i)}$. ■

Propositions 6 and 7 show that both $E_{PSM}^{(i)}$ and $E_{NO\_PSM}^{(i)}$ are functions of $T^{(i)}$ and $T_{ac}^{(i)}$. Therefore, we now focus on characterizing these two quantities.

7.4.1.1. Modeling the download interval

Provided a generic i-th replica, $T^{(i)}$ is composed by two factors (see Figure 7.3): (i) the total time during which bursts are downloaded ($T_{data}^{(i)}$), and (ii) the total time during which the application at the tagged mobile host does not exchange data on the network ($T_{idle}^{(i)}$). By denoting with $td_k^{(i)}$ the time required by the tagged mobile host to download the k-th burst in the i-th replica, the following Lemma holds.

Lemma 3 *The length in time of the i-th download interval is*

$$T^{(i)} = T_{data}^{(i)} + T_{idle}^{(i)} = \sum_{k=1}^{N_{BR}^{(i)}} td_k^{(i)} + \sum_{k=1}^{N_{BR}^{(i)}} UTT_k^{(i)} \ . \tag{7.4}$$

Furthermore, the sets $\left\{T_{data}^{(i)}\right\}_i$, $\left\{T_{idle}^{(i)}\right\}_i$ and $\left\{T^{(i)}\right\}_i$ are composed by identically distributed random variables.

Proof. Equation 7.4 follows immediately by the definitions of $N_{BR}^{(i)}$, $td_k^{(i)}$, $UTT_k^{(i)}$, and download interval. To prove that the r.v. $\left\{T^{(i)}\right\}_i$ are i.d., we can procede as follows. We have assumed that the TCP connection supporting the burst download is in steady state. This means that it can be described by a stationary stochastic process, and, hence, its statistics can be assumed to be i.d. among different burst downloads. Furthermore, by definition, the burst sizes are i.i.d. random variables. Since the time required to download the k-th burst (i.e., $td_k^{(i)}$) depends on the TCP-connection statistics and on the burst size, the random variables $\left\{td_k^{(i)}\right\}_k$ are identically distributed. Moreover, the number of bursts downloaded in each replica (i.e., $N_{BR}^{(i)}$) is sampled from i.i.d. random variables, and hence the random variables $\left\{T_{data}^{(i)}\right\}_i$ are i.d. At the same time, the lengths in time of User Think Times are sampled from i.i.d. random variables, and hence the random variables $\left\{T_{idle}^{(i)}\right\}_i$ are i.d. as well. Therefore, we can conclude that also the random variables $\left\{T^{(i)}\right\}_i$ are identically distributed. ■

Lemma 3 guarantees that the average values of T_{data}, T_{idle} and T there exist. As a preliminary step to evaluate a closed form of $E\left[T\right]$, it is worth evaluating the average value of the time required to download a single burst, i.e. $E\left[td\right]$.

To characterize the behavior of the TCP connection during burst donwloads – and, ultimately, to evaluate $E\left[td\right]$ – we use the following line of reasoning. If we neglect the probability of duplicated TCP segments to be still flowing after the end of a burst download, we can assume that

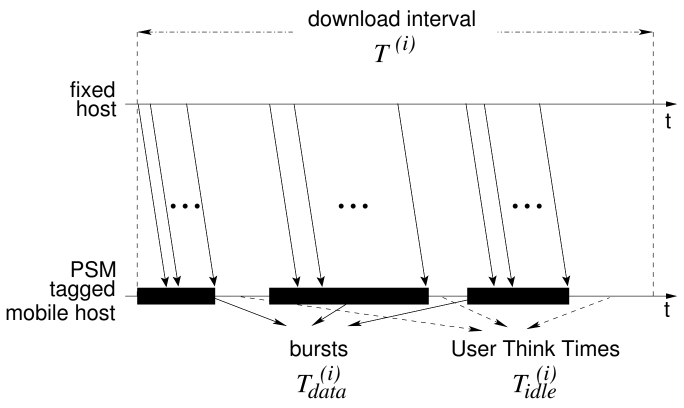


Figure 7.3.: data-transfer and idle phases in the i-th download interval

the TCP connection is completely “frozen" during the User Think Time between two consecutive burst downloads. Hence, as far as the TCP behavior, we can consider an equivalent version of the application-level traffic, where bursts are downloaded *continuously*, without being interleaved by User Think Times. By replicating the download of a burst for an arbitrary number of times, say h, we can define the random variables $\{\gamma_h\}_{h>0}$ as follows:

$$\gamma_h \triangleq \frac{\sum_{q=1}^{h} d_q}{\sum_{q=1}^{h} td_q} \;, \tag{7.5}$$

where the random variables d_q and td_q represent the size of the q-th burst and the time required to download it, respectively. For a particular value of h, the random variable γ_h is the throughput achieved by the TCP connection up to the download of the h-th burst. Since we have assumed that the TCP connection is in steady state, $\{\gamma_h\}_{h>0}$ is a stationary stochastic process, whose average value is the throughput of the TCP connection, throughout referred to as γ_{TCP}. Based on these observations, the following lemma holds.

Lemma 4 *The throughput offered by the TCP connection is*

$$\gamma_{TCP} = \frac{E[d]}{E[td]} \;, \tag{7.6}$$

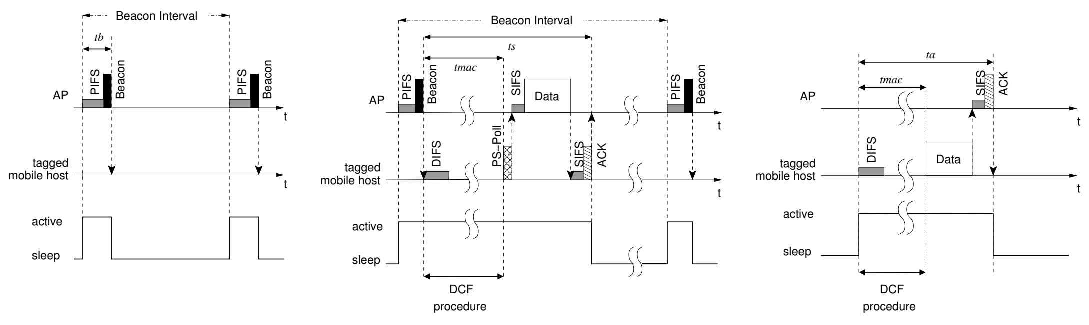
where $E[d]$ and $E[td]$ represent the average size of bursts, and the average time required to download them, respectively.

Proof. Since $\{\gamma_k\}_k$ is a stationary process, we can evaluate its average value (i.e., γ_{TCP}) for any h. Specifically, we can evaluate it in the limit of $h \to \infty$. By exploiting Equation 7.5 we can derive γ_{TCP} as follows:

$$\gamma_{TCP} = E\left[\lim_{h\to\infty} \frac{\sum_{q=1}^{h} d_q}{\sum_{q=1}^{h} td_q}\right] = E\left[\lim_{h\to\infty} \frac{\frac{\sum_{q=1}^{h} d_q}{h}}{\frac{\sum_{q=1}^{h} td_q}{h}}\right] = E\left[\frac{E[d]}{E[td]}\right] = \frac{E[d]}{E[td]} \;.$$

■

Based on Lemma 4, $E[td]$ is equal to $E[d]/\gamma_{TCP}$. This result, together with Lemma 3, allows us to derive a closed form of $E[T]$. Specifically, Theorem 5 holds.



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

Figure 7.4.: PSM analysis.

Theorem 5 *The average length of a download interval is*

$$E\left[T\right] = E\left[T_{data}\right] + E\left[T_{idle}\right] = E\left[N_{BR}\right] \cdot \left\{ \frac{E\left[d\right]}{\gamma_{TCP}} + E\left[UTT\right] \right\} . \tag{7.7}$$

Proof. Based on Lemma 3 the random variables $\left\{T_{data}^{(i)}\right\}_i$ and $\left\{T_{idle}^{(i)}\right\}_i$ are identically distributed. Therefore, since an average value of T_{data} and T_{idle} there exists, $E\left[T\right]$ can be expressed as $E\left[T_{data}\right] + E\left[T_{idle}\right]$. By definition, $T_{data}^{(i)}$ is equal to $\sum_{k=1}^{N_{BR}^{(i)}} td_k^{(i)}$. Furthermore, the random variables $\left\{N_{BR}^{(i)}\right\}_i$ and $\left\{td_k\right\}_k$ are mutually independent. Thus, $E\left[T_{data}\right]$ is equal to $E\left[N_{BR}\right] \cdot E\left[td\right]$. On the other hand, $T_{idle}^{(i)}$ is equal to $\sum_{k=1}^{N_{BR}^{(i)}} UTT_k^{(i)}$. Since the random variables $\left\{N_{BR}^{(i)}\right\}_i$ and $\left\{UTT_k\right\}_k$ are mutually independent, $E\left[T_{idle}\right]$ is equal to $E\left[N_{BR}\right] \cdot E\left[UTT\right]$. The last formulation of Equation 7.7 derives immediately from Lemma 4. ■

7.4.1.2. Modeling the time spent in the active mode

In this section we focus on the time spent in the active mode by the PSM tagged mobile host while downloading the bursts of a generic i-th replica, i.e., $T_{ac}^{(i)}$.

According to the assumption of using a TCP/IP architecture, the traffic on the WLAN related to the tagged mobile host results from three components: i) the TCP segments[3] coming from the server; ii) the TCP acks sent to the server; and iii) the Beacon frames sent by the Access Point. Thus, $T_{ac}^{(i)}$ is the time spent in the active mode by the tagged mobile host to handle these traffic components. At this point, it is worth discussing some assumption and property of the reference scenario upon which the following analysis is based. As far as the Beacon frames, in the following we assume that the mobile hosts in the hotspot freeze at the beginning of each Beacon Interval, waiting for receiving a Beacon frame from the Access Point. This assumption is aligned with the most up-to-date proposals within the 802.11 working groups [36]. Hence, Beacon frames never undergo collisions, and they are correctly received by every mobile host in the hotspot (Figure 7.4(a)). As far as the TCP segments, it is worth noting that the tagged mobile host downloads each TCP

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

segment from the Access Point inside a distinct Data frame. It should be noted that we assume that the RTS/CTS mechanism is disabled. Hence, due to the 802.11 specification [35], such downloads occur by exchanging (between the Access Point and the tagged mobile host) a sequence of frames[4] composed by a PS-Poll, Data, and Ack frame, as shown in Figure 7.4(b). Similarly, each TCP ack is uploaded to the Access Point inside a distinct Data frame, as shown in Figure 7.4(c). In this case i) the tagged mobile host transmits the Data frame by using the standard DCF procedure, and ii) after a SIFS interval the Access Point transmits the corresponding Ack frame. Based on these observations, the following lemma holds.

Lemma 5 *Let i) $ts_j^{(i)}$ be the time required by the tagged mobile host to download the j-th TCP segment during the i-th replica, starting from the point in time when the tagged mobile host starts the DCF procedure in order to send the related PS-Poll frame; ii) $ta_r^{(i)}$ 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 in order to send the related Data frame; and iii) $tb_l^{(i)}$ be the time required by the tagged mobile host to receive the l-th Beacon frame during the i-th replica. Then, for any triple $\langle j, r, l\rangle$, the time intervals ts_j, ta_r and tb_l do not overlap.*

Proof. The time intervals during which Beacon frames are sent by the Access Point do not overlap with time intervals related to any other frame, since we have assumed that every mobile host freezes at the beginning of each Beacon Interval. Furthermore, both in the case of a TCP-segment download, and in the case of a TCP-ack upload, the first frame of the respective sequence is sent by the tagged mobile host (see Figure 7.4). Therefore, the time intervals during which TCP segments are downloaded and TCP acks are uploaded do not overlap with each other. ■

Based on Lemma 5, we can express $T_{ac}^{(i)}$ as in the following proposition.

Proposition 8 *The total time during which the tagged mobile host is active during the i-th replica is*

$$T_{ac}^{(i)} = \sum_{j=1}^{N_{seg}^{(i)}} ts_j^{(i)} + \sum_{r=1}^{N_{ack}^{(i)}} ta_r^{(i)} + \sum_{l=1}^{N_b^{(i)}} tb_l^{(i)} \ , \tag{7.8}$$

where $N_{seg}^{(i)}$, $N_{ack}^{(i)}$ and $N_b^{(i)}$ are the number of TCP segments downloaded by the tagged mobile host, the number of TCP acks sent by the tagged mobile host, and the number of Beacon frames sent by the Access Point during the i-th replica, respectively.

Proof. Since ts_j, ta_r and tb_l do not overlap, the total time during which the tagged mobile host is active during a download interval is the sum of all time intervals during which it receives Beacon frames, during which it downloads TCP segments, and during which it uploads TCP acks. Based on this observation, deriving Equation 7.8 is straightforward. ■

Equation 7.8 shows that $T_{ac}^{(i)}$ is composed by three factors, i.e., the time related to downloading the TCP segments, the time related to uploading TCP acks, and the time related to receiving Beacon frames. To highlight this dependence more clearly, we exploit the following definition.

[4]A sequence of frames is a set of frames spaced by SIFS intervals. Hence, just the first frame of in a sequence may undergo collision.

Definition 1

- $T_{seg}^{(i)}$ *is the total time during which the tagged mobile host is active to download the TCP segments during the i-th replica, i.e.,* $T_{seg}^{(i)} = \sum_{j=1}^{N_{seg}^{(i)}} ts_j^{(i)}$;
- $T_{ack}^{(i)}$ *is the total time during which the tagged mobile host is active to upload the TCP acks during the i-th replica, i.e.,* $T_{ack}^{(i)} = \sum_{r=1}^{N_{ack}^{(i)}} ta_r^{(i)}$;
- $T_b^{(i)}$ *is the total time during which the tagged mobile host is active to receive the Beacon frames during the i-th replica, i.e.,* $T_b^{(i)} = \sum_{l=1}^{N_b^{(i)}} tb_l^{(i)}$.

Based on Definition 1, $T_{ac}^{(i)}$ can be written as:

$$T_{ac}^{(i)} = T_{seg}^{(i)} + T_{ack}^{(i)} + T_b^{(i)} \ . \tag{7.9}$$

In the following of the section we separately analyze each of these terms. Specifically, we show that the sets $\left\{T_{seg}^{(i)}\right\}_i$, $\left\{T_{ack}^{(i)}\right\}_i$, and $\left\{T_b^{(i)}\right\}_i$ are composed by i.d. random variables. By deriving their average values, we eventually provide a closed form of $E\left[T_{ac}\right]$.

Let us start by analyzing $T_b^{(i)}$. To this end, Lemma 6 provides a closed form of $tb_l^{(i)}$.

Lemma 6 *Let*

- t_{sa} *the time required by the tagged mobile host to switch from the sleep to the active mode;*
- $PIFS$ *be the length in time of a PIFS interval;*
- τ *be the propagation delay between the tagged mobile host and the Access Point;*
- $phyHdrSz$ *be the size in bits of the physical-level header of an 802.11 frame;*
- $phyR$ *and* $baseR$ *be the rate at which the physical and MAC-level headers are transmitted, respectively;*
- $beacSz_l^{(i)}$ *be the size in bits of the l-th Beacon frame sent by the Access Point during the i-th replica, excluding the physical header.*

Then the time during which the tagged mobile host is active to receive the l-th Beacon frame during the i-th replica is

$$tb_l^{(i)} = t_{sa} + PIFS + \tau + \frac{phyHdrSz}{phyR} + \frac{beacSz_l^{(i)}}{baseR} \ . \tag{7.10}$$

Proof. In our model we assume that the tagged mobile host is sleeping at the end of a Beacon Interval. In other words, we assume that the traffic buffered at the Access Point at the beginning of a Beacon Interval is handled by the tagged mobile host within the Beacon Interval itself. Hence, to receive the next Beacon frame, the tagged mobile host has to switch to the active mode. Then, if we assume that the Access Point and the tagged mobile host are synchronized, the tagged mobile host remains active for the time required by the Access Point to transmit a Beacon frame. By recalling which fields compose a Beacon frame, and at which data rate each field is transmitted [35], it is straightforward deriving Equation 7.10. ■

It is worth noting that, in principle, $beacSz_l^{(i)}$ is a random variable, as the length of the TIM is variable (see [35]). However, for the sake of simplicity, in the following we assume that the length of the TIM (and thus the size of the Beacon frame) are constant. Therefore, hereafter tb denotes the (constant) time interval during which the tagged mobile host is active to receive a Beacon frame. We are now in the position of completing the characterization of $T_b^{(i)}$. Specifically, the following lemma holds.

Lemma 7 *The random variables $\left\{T_b^{(i)}\right\}_i$ are i.d.. Moreover, the average value of T_b is*

$$E\left[T_b\right] = \frac{E\left[T\right]}{BI} \cdot tb\ , \tag{7.11}$$

where $E\left[T\right]$ is the average value of the download interval (see Equation 7.7), and BI is the length in time of a Beacon Interval.

Proof. By recalling its definition, $T_b^{(i)}$ can be written as $T_b^{(i)} = N_b^{(i)} \cdot tb$. Furthermore, the number of Beacon Intervals occurring in a download interval can be evaluated as the ratio between the lengths in time of the download interval, and of the Beacon Interval, i.e., $N_b^{(i)} = T^{(i)} / BI$. From Lemma 3, the random variables $\left\{T^{(i)}\right\}_i$ are i.d., and hence the same property holds also for the r.v. $\left\{T_b^{(i)}\right\}_i$. The derivation of Equation 7.11 is then straightforward. ■

To complete the analysis of $T_{ac}^{(i)}$ we have now to characterize the impact of the TCP traffic, i.e., $T_{seg}^{(i)}$ and $T_{ack}^{(i)}$. Let us start by analyzing $ts_j^{(i)}$ and $ta_r^{(i)}$. It is worth recalling that in our model each TCP segment (ack) is downloaded (uploaded) in a distinct MAC-level Data frame. Therefore, $ts_j^{(i)}$ ($ta_r^{(i)}$) represents the time required, at the MAC level, to complete the frame exchange required by the tagged mobile host to download (upload) a Data frame from (to) the Access Point. By focusing on Figure 7.4 it can be noted that $ts_j^{(i)}$ starts when the tagged mobile host invokes the DCF procedure for the first transmission attempt of the j-th PS-Poll, and finishes when the tagged mobile host has received the corresponding Ack frame. On the other hand, $ta_r^{(i)}$ starts when the tagged mobile host invokes the DCF procedure for the first transmission attempt of the Data frame containing the r-th TCP ack, and finishes when the PSM mobile host has received the corresponding Ack frame. Based on these observations, and by recalling the way TCP and PSM work, we can prove the following lemma.

Lemma 8 *For each j and for each r, $ts_j^{(i)}$ and $ta_r^{(i)}$ start at the beginning of the first free slot next to the successful delivery of a frame. Therefore, the sets $\left\{ts_j^{(i)}\right\}_{i,j}$ and $\left\{ta_r^{(i)}\right\}_{i,r}$ are composed by i.d. random variables.*

Proof. In our model a new PS-Poll frame is placed in the tagged mobile host sending queue in two cases, i.e., i) after successfully receiving a Beacon frame; and ii) after successfully receiving a Data frame (containing a previous TCP segment) with the More Data Bit set. On the other hand, a new Data frame containing a TCP ack is scheduled for transmission after successfully receiving a TCP segment[5]. Therefore, it can be noted that the tagged mobile host schedules for transmission a PS-Poll frame either after receiving a Beacon frame, or after successfully downloading a previous

[5]We assumed TCP Reno without delayed acks.

TCP segment, or after successfully uploading a TCP ack. On the other hand, the tagged mobile host schedules a Data frame containing a TCP ack either after successfully downloading a TCP segment, or after successfully uploading a previous TCP ack. In any case, $ts_j^{(i)}$ and $ta_r^{(i)}$ start at the beginning of the first free slot next to the successfully delivery of a frame. Finally, by recalling the CSMA algorithm used by 802.11 networks, it is straightforward deriving that the sets $\left\{ts_j^{(i)}\right\}_{i,j}$ and $\left\{ta_r^{(i)}\right\}_{i,r}$ are composed by i.d. random variables for any possible value of j, r, and i. ■

In order to derive the average value of ts and ta, it is worth pointing out that both of them are made up of two components: i) the MAC delay experienced before the successful delivery of the first frame in the respective sequence (i.e., either the PS-Poll or the Data frame containing the TCP ack); and ii) the time required to transmit the frame sequence. In detail, the MAC delay is defined as the time interval elapsed from the point in time when the DCF procedure is invoked to transmit a frame, up to the point in time when the successfull transmission of that frame starts. Based on these observations, $ts_j^{(i)}$ and $ta_r^{(i)}$ can be expressed as in the following proposition.

Proposition 9 *Let*

- ✧ *$tmac_j^{(i)}$ and $tmac_r^{(i)}$ be random variables measuring the MAC delay experienced by the tagged mobile host to download the j-th TCP segment and upload the r-th TCP ack within the i-th replica, respectively;*
- ✧ *$TcpSegSz_j^{(i)}$ be the size in bits of the j-th TCP segment in the i-th replica;*
- ✧ *$TcpAckSz$ be the size in bits of a TCP ack;*
- ✧ *o_{dl} and o_{ul} be the overheads in time introduced by the MAC protocol to deliver the frames in the PS-Poll/Data/Ack sequence, and in the Data/Ack sequence, respectively[6];*
- ✧ *$dataR$ be the rate used to transmit the DATA field of Data frames.*

Then o_{dl} and o_{ul} can be expressed as:

$$\left\{\begin{array}{lcl} o_{dl} & = & 3\cdot\frac{phyHdrSz}{phyR}+\frac{pollSz+macHdrSz+ackSz}{baseR}+\frac{fcsSz}{dataR}+2\cdot SIFS+3\cdot\tau \\ o_{ul} & = & 2\cdot\frac{phyHdrSz}{phyR}+\frac{macHdrSz+ackSz}{baseR}+\frac{fcsSz}{dataR}+SIFS+2\cdot\tau \end{array}\right. , \tag{7.12}$$

and $ts_j^{(i)}$ and $ta_r^{(i)}$ can be expressed as:

$$\left\{\begin{array}{lcl} ts_j^{(i)} & = & tmac_j^{(i)}+o_{dl}+\frac{TcpSegSz_j^{(i)}}{dataR} \\ ta_r^{(i)} & = & tmac_r^{(i)}+o_{ul}+\frac{TcpAckSz}{dataR} \end{array}\right. . \tag{7.13}$$

Proof. Based on Lemma 8, $ts_j^{(i)}$ and $ta_r^{(i)}$ start when the tagged mobile host invokes the DCF procedure to send the first frame in the respective sequences. By definition, the time spent in the DCF procedure is the MAC delay. In our model, we assume that MAC-level frames are successfully delivered within the maximum number of retransmissions. Hence, when the DCF procedure is completed (i.e., after a MAC delay), the tagged mobile host and the Access Point exchange the frames related to either TCP segment or the TCP ack. The time required to exchange these frame

[6] o_{dl} and o_{ul} include the (physical, MAC) headers' and trailers' transmission times, and the SIFS intervals.

sequences has a fixed component (o_{dl}, o_{ul}), related to the way the MAC and PHY layers transmit the frames, and a variable component, related to the size of the Data-frame payloads involved in the sequence. Let us focus on the TCP-segment frame sequence (see Figure 7.4(b)). As far as the fixed component, it should be noted that the sequence consists in three frames, resulting in three PHY headers ($phyHdrSz$) being sent at the PHY rate ($phyR$). The PS-Poll frame ($pollSz$) and the ACK frame ($ackSz$) do not contain DATA fields, and thus they are sent at the rate used to send the headers of MAC frames ($baseR$). Obviously, the same rate is also used for the MAC header of the DATA frame ($macHdrSz$). Furthermore, the FCS field ($fcsSz$) of the DATA frame is sent at the same rate used for the DATA field ($dataR$). Finally, the DATA and ACK frames are spaced by SIFS intervals from the previous frames, and all frames require τ seconds to be propagated from the sender to the receiver. As far as the variable component, it corresponds to the time required to transmit the Data-frame payload ($TcpSegSz_j^{(i)}$, transmitted at $dataR$). These observations allow to derive the closed forms of o_{dl} and $ts_j^{(i)}$. A similar line of reasoning can be used to derive the closed form of o_{ul} and $ta_r^{(i)}$. ■

In principle, $TcpSegSz_j^{(i)}$ is a random variable. However, in our model we assume that the size of TCP segments is constant, and equal to the Maximum Segment Size (MSS). Hence, hereafter $TcpSegSz$ denotes the constant size of a TCP segment. Furthermore, based on Lemma 8, it can be shown that the MAC delays experienced by the tagged mobile host to send either PS-Poll frames of Data frames containing TCP ack are sampled from i.d. random variables. Therefore, to derive $E\left[ts\right]$ and $E\left[ta\right]$, it is sufficient deriving the average value of $tmac$. For the reader convenience, we postpone the detailed derivation to the Appendix E, which provides the closed form expression of $E\left[tmac\right]$. We are now in the position of deriving closed form expressions of $E\left[T_{seg}\right]$ and $E\left[T_{ack}\right]$. Specifically, the following lemma holds.

Lemma 9 *The sets $\left\{T_{seg}^{(i)}\right\}_i$ and $\left\{T_{ack}^{(i)}\right\}_i$ are composed by i.d. random variables. Furthermore, the average values of T_{seg} and T_{ack} are as follows:*

$$\left\{\begin{array}{lcl} E\left[T_{seg}\right] & = & \frac{E[N_{BR}]\cdot E[d]}{MSS}\cdot E\left[ts\right] \\ E\left[T_{ack}\right] & = & \frac{E[N_{BR}]\cdot E[d]}{MSS}\cdot E\left[ta\right] \end{array}\right. , \tag{7.14}$$

where $E\left[N_{BR}\right]$ is the average number of burst downloaded in a replica, and $E\left[d\right]$ is the average burst size.

Proof. By definition, $T_{seg}^{(i)}$ is equal to $\sum_{j=1}^{N_{seg}^{(i)}} ts_j$, where $N_{seg}^{(i)}$ is the number of TCP segments downloaded by the tagged mobile host during the i-th replica. By recalling that in our model the TCP segments have constant size, equal to MSS, and by assuming that the tagged mobile host does not download duplicated TCP segments, $N_{seg}^{(i)}$ is equal to $\sum_{k=1}^{N_{BR}^{(i)}} d_k \Big/ MSS$. Since, by definition, the sets $\left\{N_{BR}^{(i)}\right\}_i$ and $\left\{d_k^{(i)}\right\}_{i,k}$ are composed by i.i.d. random variables, the random variables $\left\{N_{seg}^{(i)}\right\}_i$ are i.i.d. as well. Based on this observation, and by recalling that the random variables $\left\{ts_j^{(i)}\right\}_{i,j}$ are i.d. (see Lemma 8), we can conclude that the random variables $\left\{T_{seg}^{(i)}\right\}_i$ are i.d. as well. The average value of T_{seg} can be derived by noting that i) for each couple $\langle i,j\rangle$, the random variables $N_{seg}^{(i)}$ and $ts_j^{(i)}$ are mutually independent; and ii) for each couple $\langle i,k\rangle$, the random variables $N_{BR}^{(i)}$ and $d_k^{(i)}$ are mutually independent. Finally, the closed form of $E\left[T_{ack}\right]$ can be derived by following the same line of reasoning, and by recalling that in our model $N_{ack}^{(i)}$ is equal to $N_{seg}^{(i)}$ since i) we have assumed the TCP-Reno version without delayed acks; and ii) the tagged mobile host does not download duplicated TCP segments. ■

Finally, based on Equation 7.9, Lemma 7, and Lemma 9 we can derive the closed form of $E\left[T_{ac}\right]$. Specifically, the following theorem holds.

Theorem 6 *The random variables $\left\{T_{ac}^{(i)}\right\}_i$ are identically distributed. Furtermore, the average value of T_{ac} is:*

$$E\left[T_{ac}\right]=\frac{E\left[N_{BR}\right]\cdot E\left[d\right]}{MSS}\cdot\left(E\left[ts\right]+E\left[ta\right]\right)+\frac{E\left[T\right]}{BI}\cdot tb \tag{7.15}$$

Proof. Based on Equation 7.9, Lemma 7 and Lemma 9, the random variables $\left\{T_{ac}^{(i)}\right\}$ are the sum of i.d. random variables, and hence they are identically distributed. The closed form of $E\left[T_{ac}\right]$ can be derived immediately from Equations 7.9, 7.11 and 7.14. ■

Theorems 5 and 6 allow to complete our analysis. Specifically, Theorem 7 provides the closed form expressions of E_{NO_PSM} and E_{PSM}.

Theorem 7 *The random variables $\left\{E_{NO\_PSM}^{(i)}\right\}_i$ and $\left\{E_{PSM}^{(i)}\right\}_i$ are composed by i.d. random variables. Furthermore, the following equations hold:*

$$\begin{cases} E_{NO\_PSM} & = & E\left[T\right]\cdot P_{ac} \\ E_{PSM} & = & E\left[T_{ac}\right]\cdot\left(P_{ac}-P_{sl}\right)+E\left[T\right]\cdot P_{sl} \end{cases}. \tag{7.16}$$

Proof. The sets $\left\{E_{NO\_PSM}\right\}_i$ and $\left\{E_{PSM}\right\}_i$ are composed by i.d. random variables, as $E_{NO\_PSM}^{(i)}$ and $E_{PSM}^{(i)}$ can be expressed as functions of i.d. random variables (see Equations 7.2 and 7.3). Furthermore, the closed forms of E_{PSM} and E_{NO_PSM} can be derived immediately by Propositions 6 and 7, and by Theorems 5 and 6. ■

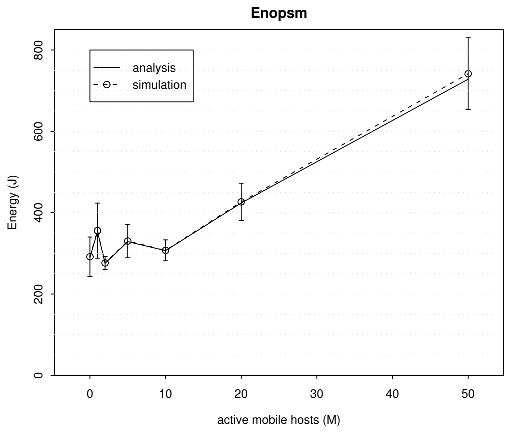
7.4.2. Model Validation

The validation of our analysis is carried out by comparing the results obtained from Equation 7.16 with the output of the simulation model described in Section 7.2. As far as the simulation parameters, we used the values reported in Table 7.2. For the reader convenience, in the Table we have also reported the average size of the bursts (i.e., $E\left[d\right]$), and the average length of the User Think Times (i.e., $E\left[UTT\right]$). As highlighted in Section 7.2, in the simulation model burst sizes and UTT lengths are sampled from the distributions provided by [19]. Hence, the values reported in the Table are the average values of these distributions.

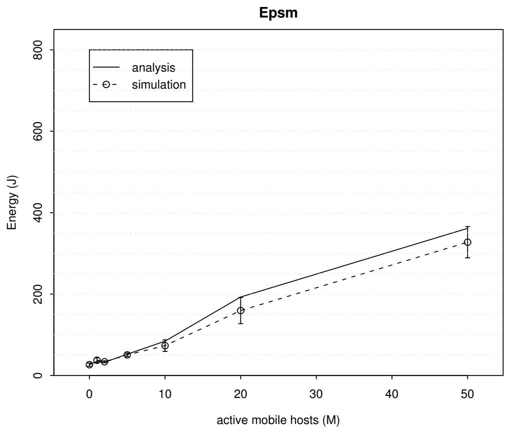
Figure 7.5 compares the analytical-model predictions with the simulation-model outcomes. Specifically, Figure 7.5 plots E_{NO_PSM} and E_{PSM} as functions of the number of active mobile hosts in the hotspot, i.e., M. As expected, the energy consumption increases with M in both cases, even though activating the Power-Saving Mode always results in lower energy consumption for a given value of M. We will come back on this point later in the analysis. At the moment, the main result of Figure 7.5 is the accuracy of our analytical model. Based on this validation, in the following we exploit both the analytical and the simulation models to understand the performance of PSM.

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

Table 7.2.: Simulation parameters used to validate the analytical model



(a) E_{NO_PSM}



(b) E_{PSM}

Figure 7.5.: Validation plots.

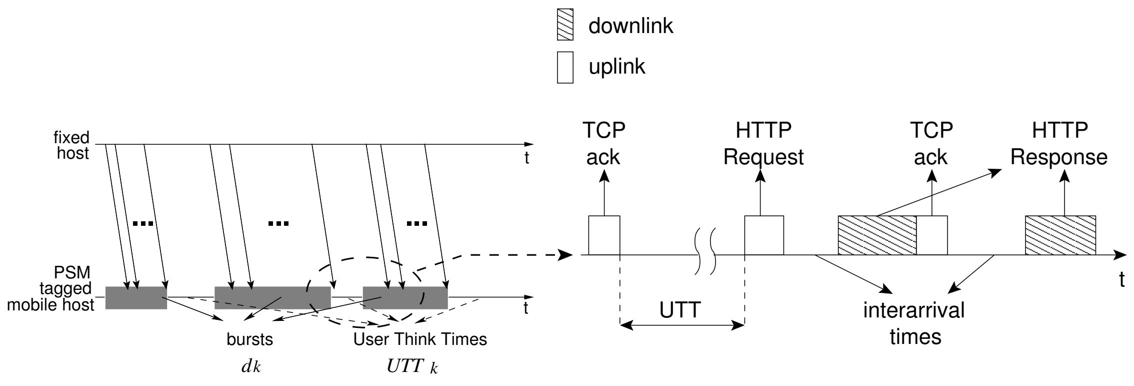


Figure 7.6.: Idle times: interarrival times and User Think Times.

7.5. Performance Evaluation of 802.11 PSM

In this section we deeply analyze the power-saving performance of PSM. To this end, in Section 7.5.1 we identify which are the sources of energy wastages within the traffic of Figure 7.1(b). This highlights with respect to which parameters PSM has to be analyzed. Then, in Sections 7.5.2 and 7.5.3 the results of our experiments are presented and discussed in depth.

7.5.1. Idle times: the problem 802.11 PSM aims to solve

The goal of any power-management strategy is reducing as much as possible the energy spent to transfer the application-level data. In the ideal case, the wireless interface of the mobile host should be active just for receiving (sending) data from (to) the fixed host(s), and should remain in a "low-power mode" (possibly, it should be switched off) whenever it becomes idle. The PSM follows this approach, operating at the MAC layer.

In a classical TCP/IP environment, PSM has to deliver the segments generated at the transport level, and minimize the energy spent during *idle times* between them. To clarify this point, let us focus on the reference traffic described in Section 7.2, whose scheme is reported in the left-hand side of Figure 7.6. Different types of idle times can be identified, one of them being *User Think Times*. A User Think Time (UTT) occurs between the last TCP segment acknowledging (the data of) a burst, and the first TCP segment related to the next burst. Clearly, User Think Times are related to the user behavior – for example, in the Web case, they are time intervals the user spends reading pages. Furthermore, if we look inside the burst-download phases, we find a second class of idle times, referred to as *interarrival times*. The right-hand side of Figure 7.6 highlights this. Interarrival times are not related to the user behavior, but are generated by the network protocols' mechanisms. Due to their different nature, User Think Times and interarrival times follow different statistical distributions [27, 25, 19, 13]. Usually, one second is chosen as the bound between the two classes [27, 25, 19]. It should be noted that the User Think Times' distribution is typically quite spread, since User Think Times can be as large as 60 s [27] and beyond.

Based on these remarks, the effectiveness of a power manager, and, specifically, of PSM, must be evaluated with respect to *both* kinds of idle times. In Section 7.5.2 we analyze the performance of PSM with respect to interarrival times, i.e, we study its behavior during burst-download phases. The performance of PSM during User Think Times is then analyzed in Section 7.5.3.

7.5.2. Power Management during burst-download phases

Burst-transfer phases (and interarrival times) are determined by both the application and the networking protocols. In our scenario, where data flow mainly in the downlink direction, the application(s) dictates the burst sizes[7], while TCP is the main responsible for interarrival times. Thus, we now focus on the impact of two parameters, i.e. i) the average size of bursts, and ii) the TCP-connection throughput. In particular, the impact of the average size of bursts is analyzed in Section 7.5.2.1, while Section 7.5.2.2 is devoted to analyze the dependence on TCP throughput. It should be noted that the throughput of a TCP connection is basically defined by two network parameters, i.e. the segment-loss probability (throughout referred to as p_l^{tcp}) and the Round Trip Time (RTT) [45]. Therefore, we analyze the impact of both of these parameters on PSM. The analysis carried out in sections 7.5.2.1 and 7.5.2.2 assumes a single user in the hotspot (i.e, $M = 0$). Section 7.5.2.3 extends the analysis by considering the case of congested WLANs (i.e., $M > 0$).

7.5.2.1. Varying the average size of bursts

As highlighted in Section 7.2, in our experiments bursts are represented by Web pages, and we use the statistical models presented in [19, 25] to define their size. Whenever the size of a burst has to be defined, a sample (say, s) from the distribution of Web-page sizes is drawn. This sample is then multiplied by a "scaling factor", a, and the burst size is $a \cdot s$. This way, the average value of burst sizes can be scaled (i.e., it is equal to $a \cdot \mu$, where μ is the average value of Web-page sizes), without modifying the distribution's coefficient of variation. This property is very important to evaluate PSM under realistic traffic loads. There is wide consensus on the type of distribution that best fits Web-page sizes (see, for example, [19, 25, 13]). On the other hand, the average value of this distribution can be highly variable, as it depends on the Web-page contents, and hence on the type of Web sites used for the analysis. For example, the Web pages considered in [19, 25] have an average size of about 20 KB. On the other hand, [27] shows that there is a clear trend towards bigger Web pages, basically due to the increased number of small embedded files (e.g., icons, banners, etc.) decorating the pages. Moreover, [27] shows that the distribution of Web-page sizes has an ever heavier tail, since the HTTP protocol is substituting FTP for downloading large files. Specifically, a non-negligible portion of Web pages is as big as few MB (e.g., pages including audio/video files). Based on these remarks, we report here a set of experiments where a varies between 1 and 100, while μ is set to 20 KB. This results in average burst sizes ranging from 20 KB to 2 MB, and thus the results are representative for a large spectrum of Web downloads, including small Web pages, but also large multimedia files. Results presented in this section have been

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

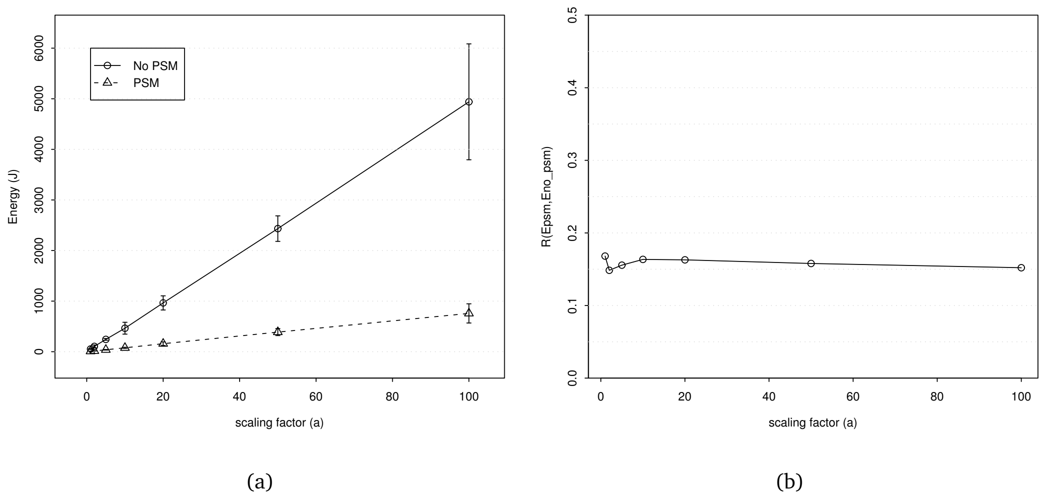


Figure 7.7.: Energy plots (a) and $R\left(E_{NO\_PSM}, E_{PSM}\right)$ (b) as functions of a.

obtained from the simulation model. Apart from $E\left[UTT\right]$ and $E\left[d\right]$, the simulation parameters are as shown in Table 7.2. As far as the UTT values, in this set of experiments User Think Times were always equal to 0. This way we can highlight the performance of PSM just during burst-download phases. Finally, to have better insights on the PSM behavior, we discuss the simulation results by exploiting the analytical expression of E_{PSM} and E_{NO_PSM} derived in Section 7.4.

Figure 7.7(a) plots E_{PSM} (bottom curve) and E_{NO_PSM} (top curve) resulting from each experiment. The most interesting feature is the linear increase of the energy in both cases. This behavior can be explained by means of Equation 7.16. Specifically, by recalling that in these experiments $E\left[UTT\right]$ is equal to 0, E_{NO_PSM} becomes as follows:

$$E_{NO\_PSM} = P_{ac} \cdot \frac{E\left[N_{BR}\right] \cdot E\left[d\right]}{\gamma_{TCP}} = P_{ac} \cdot \frac{E\left[N_{BR}\right] \cdot \mu \cdot a}{\gamma_{TCP}} \triangleq a \cdot \mu \cdot K_{NO\_PSM} \ , \tag{7.17}$$

where K_{NO_PSM} groups all the terms that are independent of both a and μ. On the other hand, E_{PSM} can be expressed as follows:

$$E_{PSM} = a \cdot \mu \cdot K_{PSM} \ , \tag{7.18}$$

where K_{PSM} groups terms that are independent of both a and μ. For ease of reading, we postpone the detailed derivation of K_{PSM} to the Appendix F. The linear increase of E_{NO_PSM} and E_{PSM} with the average burst size has an intuitive explanation. As shown in Equation 7.16 E_{NO_PSM} and E_{PSM} increase linearly with $E\left[T\right]$ and $E\left[T_{ac}\right]$. Both of these terms are proportional to $a \cdot \mu$. Specifically, as far as $E\left[T\right]$, it is worth recalling that we have assumed that the TCP connection is in steady state, and hence the TCP throughput (γ_{TCP}) is constant over time. Therefore, as shown in Equation 7.7, when $E\left[UTT\right]$ is equal to 0, the average length of the download interval is proportional to the average burst size (i.e., $a \cdot \mu$). On the other hand, $E\left[T_{ac}\right]$ results from two components, i.e., the time spent to download (upload) TCP segments (TCP acks), and the time required to receive Beacon frames from the Access Point. Clearly, the average time required to download (upload) the TCP segments (TCP acks) is proportional to the number of TCP segments

(TCP acks) in the download interval, and hence to the burst size. Furthermore, increasing the average burst size means increasing (following a proportional law) the download interval length and, ultimately the average number of Beacon frames that have to be received.

The results plotted in Figure 7.7(a) allow to show an important property of PSM, which is better highlighted in Figure 7.7(b) plotting the $R(E_{PSM}, E_{NO\_PSM})$ index[8]. The main observation about Figure 7.7(b) is that $R(E_{PSM}, E_{NO\_PSM})$ is almost independent of the average burst size. Intuitively, this property stems from the fact that the energy consumption is proportional to the burst size, either PSM is active or not. In detail, $R(E_{PSM}, E_{NO\_PSM})$ is equal to K_{PSM}/K_{NO_PSM}, and hence it is independent of $a \cdot \mu$. The specific value of $R(E_{PSM}, E_{NO\_PSM})$ depends on the parameters that define K_{PSM} and K_{NO_PSM}. In the case of our experiments, this value is around 0.16, resulting in energy saving around 84%. Based on Figure 7.7(b) we can claim that *the energy saved by PSM does not significantly depend on the average burst size*. Therefore, unless otherwise stated, the results that we present hereafter have been derived by assuming $a = 1$.

7.5.2.2. The impact of the Internet throughput

The experimental results presented in [45] show that the segment-loss probability (p_l^{tcp}) and the Round Trip Time (RTT) are the main network parameters that define the throughput of a TCP connection (γ_{TCP}). Specifically, if we assume that no timeouts occur, the relationship among p_l^{tcp}, RTT, and γ_{TCP} is as follows:

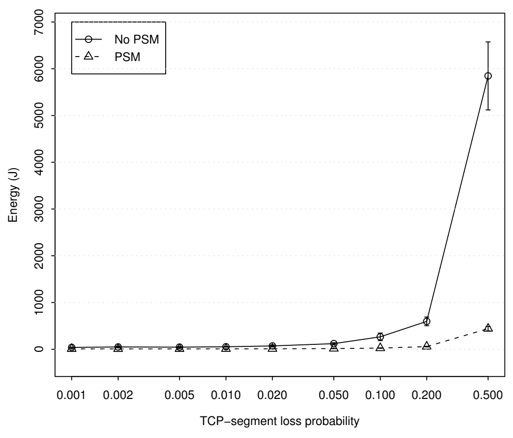
$$\gamma_{TCP} \propto \frac{1}{RTT \cdot \sqrt{p_l^{tcp}}} \, . \tag{7.19}$$

When the effect of timeouts is included, the above equation becomes more complex. However: i) p_l^{tcp} and RTT still remain the network parameters defining γ_{TCP}, and ii) γ_{TCP} still remains a decreasing function of both. Thus, we ran a set of simulation experiments to analyze how PSM behaves with respect to both p_l^{tcp} and RTT.

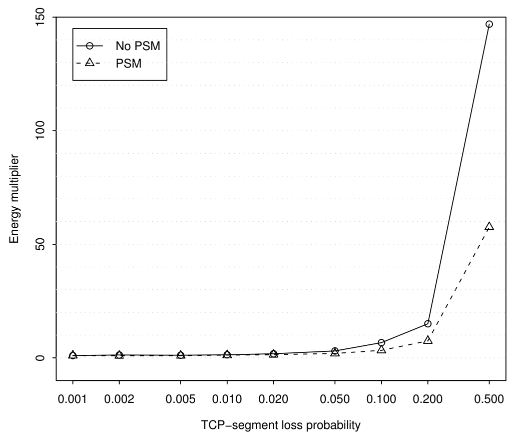
Figure 7.8(a) plots E_{PSM} (bottom curve) and E_{NO_PSM} (top curve) as functions of p_l^{tcp}. In our experiments, the lower and upper bounds of p_l^{tcp} were set to 0.001 and 0.5, respectively (according to [45]). The rest of the simulation parameters were set as in Table 7.2, unless $E[UTT]$, which was set to 0. As expected, both E_{PSM} and E_{NO_PSM} increase with p_l^{tcp}. Intuitively, increasing p_l^{tcp} means reducing the TCP throughput. Therefore, the average length of the download interval ($E[T]$) increases, and this results in an increase of both E_{NO_PSM} and E_{PSM}, as discussed in the previous section. This behavior can be understood by relying again on the analytical results presented in Section 7.4. For ease of reading the detailed proof of this claim is provided in the Appendix G.

The most interesting feature of Figure 7.8(a) is that the rate of increase of E_{PSM} is far lower than the rate of E_{NO_PSM}. To quantify this feature, let us focus on Figure 7.8(b). If $E(\hat{p})$ denotes the energy spent when p_l^{tcp} is equal to $\hat{p}$, Figure 7.8(b) plots the ratio between $E(\hat{p})$ and $E(0.001)$. Therefore, it shows the multiplicative factors with respect to the minimum power consumption,

[8]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 also shows the energy saved by PSM.



(a) Energy plots



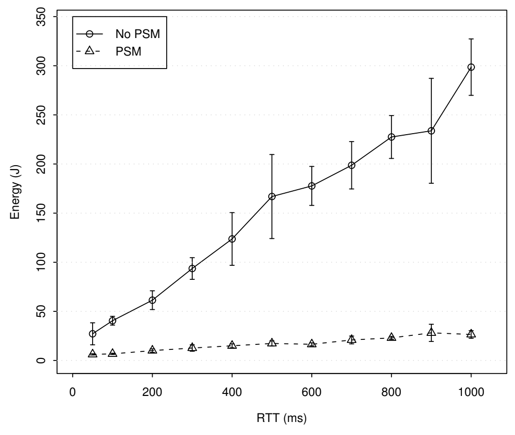
(b) Multiplicative factors

Figure 7.8.: PSM performance as function of the TCP segment-loss probability (p_l^{tcp}).

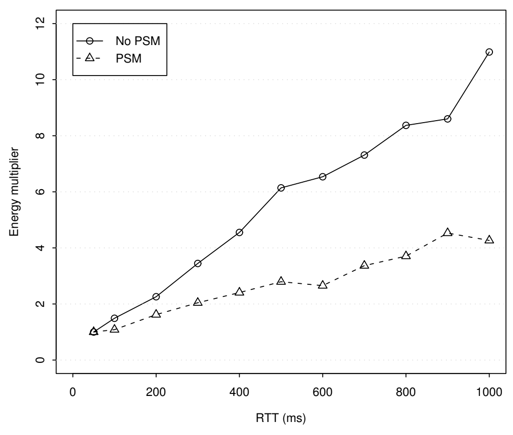
achieved at p_l^{tcp}=0.001. It is worth noting that the multiplicative factor when PSM is used is always lower than the multiplicative factor without PSM. Furthermore, the more p_l^{tcp} increases, the more the difference among the two curves increases. Qualitatively, the motivation of this behavior is that when p_l^{tcp} increases the TCP throughput decreases and, hence, the total idle time increases. When PSM is not used, the additional idle time (with respect to $p_l^{tcp} = 0.001$) is spent completely in the active mode. On the other hand, when the PSM is used, this additional idle time is only partly spent in the active mode (due to Beaconing), and mostly spent in the sleep mode. Hence, the negative effect on the power consumption is reduced. By observing Figures 7.8(a) and 7.8(b), we can conclude that PSM suffers from an increase of p_l^{tcp}. However, when it is used, the negative effect of high p_l^{tcp} values on the power consumption is highly mitigated. A similar result is also obtained when analyzing the dependence of energy consumption on RTT. The results reported in Figures 7.9(a) and 7.9(b) are obtained by varying RTT between 50 ms and 1 s. Figure 7.9(a) shows the energy expenditures (E_{PSM} and E_{NO_PSM}), while Figure 7.9(b) shows the multiplicative factors. Though the absolute values are different from those in Figures 7.8(a) and 7.8(b), the qualitative behavior is similar. The power consumption is negatively affected by large Round Trip Times, whether PSM is used or not. However, PSM helps in mitigating this effect.

7.5.2.3. Limitations due to WLAN congestion

So far, the analysis has been carried out under the assumption of a single mobile host within the Wi-Fi hotspot, i.e., M was assumed to be equal to 0. Now, we evaluate the impact of MAC-level congestion on the mobile host power consumption (i.e., $M > 0$). To this end, we firstly highlight the limitations of PSM when a standard TCP architecture is used. Then, we show to what extent an Indirect-TCP architecture [16] can alleviate these problems, and we suggest a simple MAC-level mechanism that can further help in this sense. Results presented in this section are obtained from our simulation model. The simulator parameters were set as shown in Table 7.2, apart from



(a) Energy plots



(b) Multiplicative factors

Figure 7.9.: PSM performance as function of the Round Trip Time (RTT).

$E[UTT]$, which was set to 0.

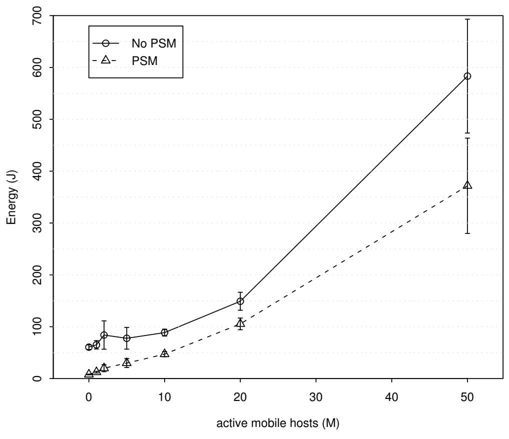
802.11 PSM in a standard TCP architecture

Figure 7.10(a) plots E_{PSM} and E_{NO_PSM} as functions of the number of active background mobile hosts in the hotspot (i.e., M). As expected, both E_{PSM} and E_{NO_PSM} increase when the congestion level on the WLAN increases. On one hand, the MAC-level congestion has a severe impact on the TCP-connection throughput. The impact of the WLAN congestion on the TCP throughput is evident from Figures 7.10(b,c,d). Specifically, the frame loss probability on the WLAN increases with M (Figure 7.10(b)). This leads to an increased number of timeouts at the TCP level (Figure 7.10(c)), and, ultimately, to a severe degradation of the TCP throughput (Figure 7.10(d)). As discussed in Section 7.5.2.2, throughput decreases have a negative impact on both E_{NO_PSM} and E_{PSM}. On the other hand, the MAC-level congestion has also a direct effect on E_{PSM}. Specifically, the analysis of the MAC delay presented in Appendix G shows that the average MAC delay increases with M. Therefore, the times required to download a TCP-segment (i.e., $E[ts]$), and to upload a TCP-ack (i.e., $E[ta]$), increase as well. Hence, the time during which the tagged mobile host is active ($E[T_{ac}]$), and, eventually, E_{PSM} increase with M (see Equations 7.15 and 7.16).

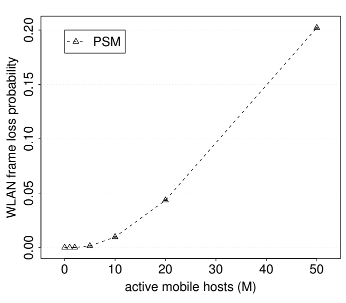
Based on these observations, two factors are responsible for the increased power consumption when M increases, i.e., i) the reduced TCP throughput (due to an increase in the frame loss probability); and ii) the increased MAC delay. In the following of this section, we decouple the effects of these factors, to understand the real impact of each one.

802.11 PSM in an Indirect TCP architecture

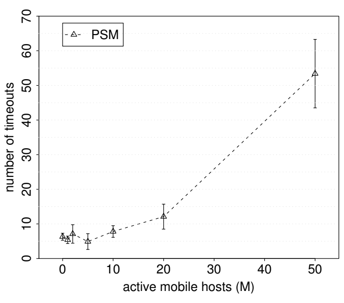
In this section we show how an Indirect TCP architecture can mitigate the energy consumption in the case of highly congested wireless LAN. For the reader convenience, we replicate the scheme of the Indirect-TCP architecture in Figure 7.11. It has been widely proved [17] that this architecture shields a TCP sender at the fixed host from the losses on the wireless link, thus increasing the throughput with respect to the standard TCP architecture. We show that this "shielding property"



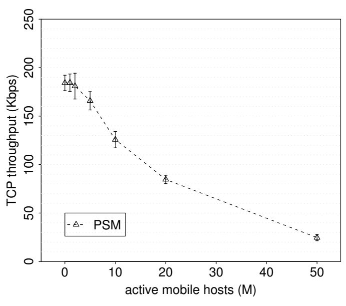
(a) Energy expenditure



(b) WLAN frame loss probability



(c) number of TCP timeouts



(d) TCP throughput

Figure 7.10.: 802.11 PSM performance in a standard TCP architecture.

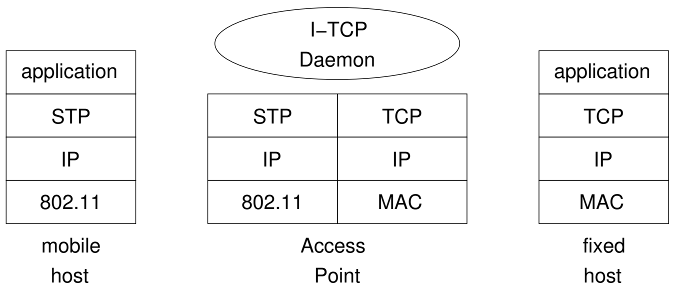
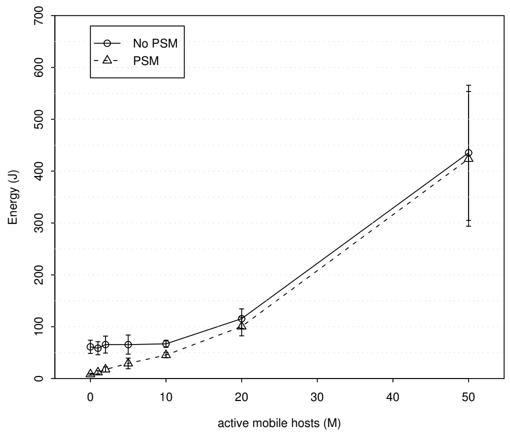


Figure 7.11.: Indirect-TCP architecture

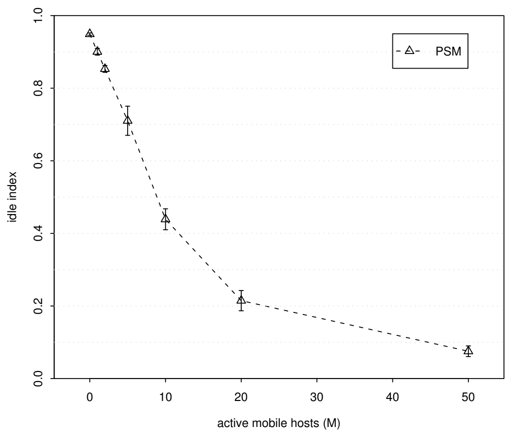
can be exploited to mitigate some of the problems related to WLAN congestion that have been highlighted before in this section. Specifically, such an architecture allows to eliminate the energy wastage related to the transport protocol, and to highlight only the effect of the increased MAC delay on energy consumption. In detail, we have replicated the simulation experiments by substituting the standard TCP architecture with the architecture shown in Figure 7.11.

Figures 7.12(c,d,e) show that the Indirect-TCP architecture actually shields the TCP-sender at the fixed host from the congestion on the WLAN (note that the TCP throughput is measured at the fixed host). Specifically, even though the WLAN packet loss probability increases as in the standard-TCP case (Figure 7.12(c)), the number of timeouts (Figure 7.12(d)) and the TCP throughput (Figure 7.12(e)) are independent of that. Hence, the effect on the additional power consumption related to the decreased TCP throughput disappears, and the MAC-delay increase is entirely responsible for the additional power consumption. PSM cannot cope with this problem, as can be seen by focusing on Figures 7.12(a,b). Specifically, Figure 7.12(b) plots the idle index as a function of M. The idle index is defined as the fraction of time (within burst-download phases) during which the tagged mobile host is idle because there are no frames buffered for it at the Access Point. When the WLAN congestion 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 more quickly than the tagged mobile host could fetch them from the Access Point. Basically, in this case the tagged mobile host is never idle, and the PSM can never switch it to the sleep mode. Thus, for high congestion levels activating the PSM or not leads to similar results (Figure 7.12(a)).

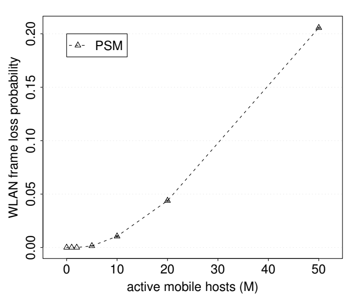
Based on these observations, we can conclude that the effect of the MAC delay can be contrasted only by reducing the MAC delay through MAC-level modifications. A promising direction is represented by the work in [21]. In this work it is shown that, in the case of concurrent downlink flows, a proper scheduling of frame delivery at the Access Point can significantly increase the channel utilization, and, hence, reduce the MAC delay with respect to the standard 802.11 protocol. Evaluating such solutions from a power-saving point of view is out of the scope of this paper. As a final remark, by focusing again on Figures 7.10(a) and 7.12(a), one could note that the E_{PSM} curves both in the standard-TCP and in the Indirect-TCP are similar. since the throughput in the TCP architecture is far lower than the throughput in the I-TCP architecture, this confirms that PSM is able to neutralize the effect of the additional download time related to low transport-level throughput.



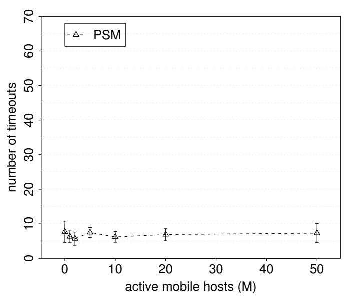
(a) Energy expenditure



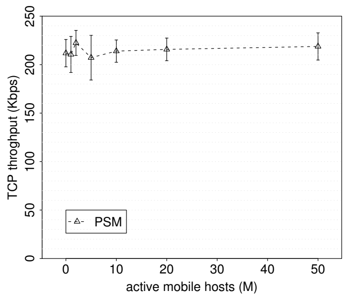
(b) Idle index



(c) WLAN frame loss probability



(d) number of TCP timeouts



(e) TCP throughput

Figure 7.12.: 802.11 PSM performance in an Indirect-TCP architecture.

To summarize, the results presented so far show that PSM works very well during burst-download phases, i.e., *it manages interarrival times very effectively*. In particular:

1. the power 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%;
2. PSM is able to limit the increase of the power consumption when the throughput offered by the TCP connection drops.

7.5.3. Is PSM effective to manage any class of idle times?

Despite the good performance shown in the previous section, one may wonder whether – and up to what extent – PSM can be improved. Since the PSM just exploits the sleep mode, one could argue that it could be improved by switching off the wireless interface instead of putting it in the sleep mode. However, this would cost additional delay and energy consumption upon re-activating the wireless interface. While the transition time from sleep to active mode (i.e., t_{sa} defined in Section 7.4) is in the order of 1 ms, the transition time from off to active mode (throughout referred to as t_{oa}) is in the order of 100 ms. It is intuitive that for "short" idle times the sleep mode should be more suitable, while for "long" idle times the best choice should be the off mode. In this section we corroborate this claim by means of the analytical model introduced in Section 7.4. This will also suggest some directions to improve the standard PSM.

Let us focus on an idle time of given length (say, t_i), and let us define the behavior of two ideal power managers, exploiting just the sleep and the off operating mode, respectively. In the ideal case, these power managers know a-priori the length of the idle time. The power manager that uses the sleep state will keep 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 power manager during t_i, the following equation holds:

$$E_S(t_i) = (t_i - t_{sa}) \cdot P_{sl} + t_{sa} \cdot P_{ac} = t_i \cdot P_{sl} + (P_{ac} - P_{sl}) \cdot t_{sa} \ . \tag{7.20}$$

On the other hand, the ideal power manager that uses the off mode will let the wireless interface in the active mode if t_i is less than t_{oa}, and will switch it off otherwise. If $E_O(t_i)$ denotes the energy spent in this case, the following equation holds:

$$E_O(t_i) = \begin{cases} t_i \cdot P_{ac} & \text{if } t_i \le t_{oa} \\ t_{oa} \cdot P_{ac} & \text{otherwise} \end{cases} \ . \tag{7.21}$$

Figure 7.13 plots Equations 7.20 ("ideal sleep" curve) and 7.21 ("ideal off" curve) as functions of t_i. The intuition claimed at the beginning of this section is confirmed: for "short" idle times the best policy is putting the wireless interface in the sleep mode, while for "long" idle times the off-based policy wins. If $\hat{t}_i$ is the crossing point between $E_S(t_i)$ and $E_O(t_i)$, the optimal (ideal) policy is a *mixed policy* that uses the sleep mode for idle times below below $\hat{t}_i$, and the off mode for idle times above $\hat{t}_i$. This simple analysis allow us to draw a fundamental bottomline for power-management

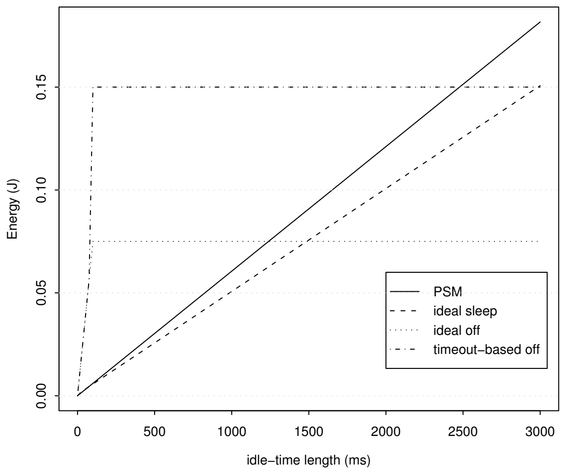


Figure 7.13.: Effectiveness of off-based and sleep-based strategies for increasing idle-time lengths.

strategies. Just using a single low-power mode is not sufficient to achieve the maximum power saving in cases where both long and short idle times are present. In those cases, mixed policies using both the sleep and the off modes should be defined.

Let us now analyze how PSM is positioned with respect to this framework. To this end, we exploit the analytical model presented in Section 7.4, and evaluate the energy spent by PSM during $\hat{t}_i$. When no data have to be exchanged with the Access Point, the tagged mobile host is active just to receive Beacons, and is sleeping for the rest of the time. By definition, the time spent in the active mode to receive a Beacon is tb. Moreover, the average number of Beacon frames sent during t seconds is $\frac{t}{BI}$, where BI is the length in time of the Beacon Interval. Thus, if $E_P(t_i)$ denotes the average energy spent by the PSM during the idle time t_i, the following equation holds:

$$E_P(t_i) = \left[t_i - \frac{t_i}{BI} \cdot tb\right] \cdot P_{sl} + \frac{t_i}{BI} \cdot tb \cdot P_{ac} = t_i \cdot \left[P_{sl} + (P_{ac} - P_{sl}) \cdot \frac{tb}{BI}\right] . \qquad (7.22)$$

Equation 7.22 is plotted in Figure 7.13, with label “PSM". This plot confirms that PSM is effective with respect to interarrival times (i.e. for idle times below 1 s). In this region, PSM is a close approximation of the ideal policy. Specifically, it outperforms the “ideal-off" policy, and the additional energy expenditure with respect to the “ideal-sleep" policy is always below 20%. This result stems from the fact that during idle times PSM activates the wireless interface just for receiving Beacon Frames.

On the other hand, PSM performs far from ideal when idle times become longer and longer, and is significantly worse than off-based policies. Just to give an idea, let us consider a very simple timeout-based policy, which lets the mobile host on for the first t_{oa} seconds of an idle time, and then switches it off[9]. The energy spent by this policy is plotted in Figure 7.13 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. Though this policy can be significantly improved [32, 3], it performs better than PSM for idle times longer than 2.5 s. For example, for

[9]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 will discuss this point in Section 7.6.

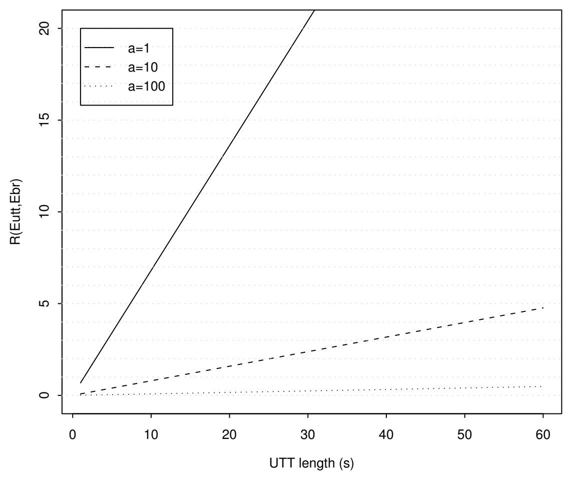
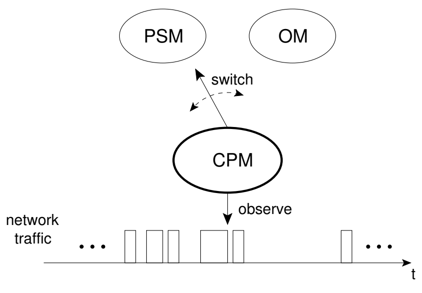


Figure 7.14.: Relative cost of User Think Times with respect to burst-download phases under 802.11 PSM.

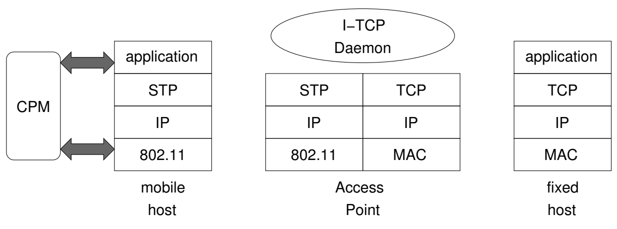
an idle time equal to 30 s, PSM spends around 12 times the energy spent by this timeout-based policy. The reason of this poor performance lies in the linear increase of $E_P(t_i)$ with respect to t_i (Equation 7.22), as compared to the constant energy expenditure of off-based policies. To summarize, the above analysis corroborates the conclusions drawn in Section 7.5.2, showing that PSM is a *near-optimal solution* to manage interarrival times, i.e., idle times below 1 s. On the other hand, with respect to User Think Times, it could be highly improved, by exploiting the wireless interface off mode.

Before analyzing in detail how such improvements can be implemented in a feasible system, let us further understand if it is actually worth reducing 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[10]. Then, the index $R(E_{UTT}, E_{BR})$ represents the average energy expenditure during User Think Times with respect to the average energy expenditure during burst-download phases. In Figure 7.14 the index $R(E_{UTT}, E_{BR})$ is plotted for increasing User Think Times. Three different plots are drawn for three different burst-size scaling factors (see Section 7.5.2.1), i.e., $a = 1$, $a = 10$, and $a = 100$. Figure 7.14 shows that the energy spent during User Think Times is not negligible with respect to the energy spent during burst-download phases. Specifically, for small bursts (i.e., $a = 1$), $R(E_{UTT}, E_{BR})$ is around 20 for UTT equal to 30 s, and raises up to around 40 for UTT equal to 60 s (out of plot). Even for large bursts (i.e., $a = 100$), the energy spent during the User Think Times is a significant fraction of the energy spent during burst-download phases. Specifically, this fraction is around 25% for UTT equal to 30 s, and raises up to around 50% for UTT equal to 60 s. This result is a strong motivation to investigate improvements of PSM focused on *long* idle times management.

[10] E_{BR} can be easily computed from the analytical results provided in Section 7.4, while E_{UTT} is equal to $E_P(E[UTT])$.



(a) Conceptual scheme



(b) Architecture

Figure 7.15.: Cross-layer Power Manager: a conceptual scheme(a) and the related architecture (b).

7.6. Enhancing the PSM: a Cross-layer Approach

One of the main outcomes of the previous section is that a *mixed* power-management policy should be used in order to improve the performance of PSM. This policy will use the standard PSM during burst-download phases, and will switch the mobile off (i.e., will use the Off Mode) during User Think Times. A conceptual scheme of a Cross-layer Power Manager (CPM) implementing such policy is shown in Figure 7.15(a) (OM denotes the Off Mode). Basically, the essence of the Cross-layer Power Manager consists in two *detection algorithms*. By observing the network traffic, the Power Manager should be able to detect both User Think Times and burst-download phases as soon as they appear. In the following of this section we propose and evaluate the detection algorithms that are taken into consideration in this paper. These algorithms exploit information about the application behavior that can be collected at the middleware layer. Therefore, CPM utilize the *cross-layer* paradigm [24], since it operates both at the middleware and at the MAC level to conserve energy (Figure 7.15(b)). Please note that, thanks to its design, CPM can be entirely implemented on the mobile host, and do not requires modifications of the wirless interface hardware. We will come back on this point later. As discussed in the following of this section, the CPM implementations we propose are simple, but nevertheless they achieve significant power saving. More sophisticated implementations could be defined, which still exploit the CPM definition.

7.6.1. Detecting burst-download phases

In a Wi-Fi hotspot environment, detecting the beginning of burst-download phases 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 assuming that *the first segment of a burst is sent by the mobile host*, i.e., it starts a request to the server[11]. Under this assumption, the beginning of a burst-download phase can be easily detected at the mobile host, and identified by the request sent by the client application after the previous User Think Time. By exploiting this, CPM can simply let the mobile

[11]In Section 7.6.4 we discuss how CPM can be extended to relax this assumption.

host in the off mode during User Think Times, and switch it in the standard PSM when a new application-level request is ready to be sent.

7.6.2. Detecting User Think Times

The next point to be addressed is how CPM detects the beginning of User Think Times. A way to deal with this problem is exploiting some knowledge about the application(s) behavior. For example, [6] presents two different power management policies implemented at the middleware layer, which are designed to support Web-based applications[12]. Both power managers rely on an agent at the mobile host, spoofing the Web traffic that the user generates. For each page, this agent is aware of the set of files composing the page itself. Once these files have been downloaded, the beginning of a User Think Time is assumed. This way of detecting User Think Times is optimal, in the sense that User Think Times are detected as soon as they start.

A first implementation of the Cross-layer Power Manager can be defined by following this approach. Specifically, a middleware agent can be included in the Cross-layer Power Manager. This agent is aware of the application that is running on the mobile host (e.g., the Web), and hence it detects User Think Times as soon as they occur. The algorithm defining the complete behavior of this Power Manager is described by the pseudo-code 1 (the Web is used as the reference application). As this implementation of the Cross-layer Power Manager depends on the particular application class it is designed for, it is hereafter referred to as the *Application-dependent* Power Manager (APM). When APM is used, the mobile host uses PSM during burst-download phases (lines 2-6), and is switched off during User Think Times (lines 7-10). The complete download of the Web page indicates that a User Think Time starts (line 6), while a new request from the user indicates that a new burst-download phase starts (line 10). The Application-dependent Power Manager uses an off-based policy to manage User Think Times. Based on Figure 7.13, one could conclude that this Power Manager performs worse than PSM for short User Think Times, i.e., around 1 s. Mechanisms to cope with this problem are not included into the Application-dependent Power Manager. This keeps its definition and its implementation straightforward. As shown in Section 7.6.3, the penalty paid in terms of power consumption is low.

The strength of the Application-dependent Power Manager relies in exploiting knowledge about the application behavior to detect the beginning of User Think Times. Hence APM can switch the mobile host off immediately once a User Think Time occurs. However, this strength may turn into a weakness. The Application-dependent Power Manager is tied with the application class it is designed for, and hence it must be changed as the application class changes. Furthermore, it could be difficult making it work in presence of concurrent applications. These drawbacks can be overcome with little performance penalties by implementing the Cross-layer Power Manager in a different way, i.e., by using an application-independent, timeout-based policy to detect User Think Times. Hereafter, this implementation of CPM is referred to as the *Timeout-based* Power Manager (TPM). The TPM design exploits the following observations. In Section 7.5.1 we have highlighted that interarrival times during burst-download phases are related to the runtime behavior of the network protocols. More precisely, in [3] it is shown that, in TCP-based scenarios, they are samples

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

```
1:  while true do
2:    switch to 802.11 PSM
3:    collect the list of files composing the Web page
4:    repeat
5:      spoof the application-level traffic
6:    until all the Web page is at the mobile host
7:    switch to Off Mode
8:    repeat
9:      <wait>
10:   until a new Web page is requested by the user
11: end while
```

Pseudo-code 1: Application-dependent Power Manager

of the Round Trip Time between the mobile and the fixed host. This observation gives a strong hint to design the Timeout-based Power Manager. The idea is to compute, on-line, a statistical characterization of the Round Trip Time. Based on this characterization, a timeout value (say, t_{TO}) is chosen, so that idle times longer than t_{TO} are very likely to be User Think Times. In other words, if at some point in time an idle time is occurring, and t_i is the time elapsed from its beginning, the equation $p\left(t_i \text{ is a UTT} | t_i \geq t_{TO}\right) \approx 1$ is assumed to hold. The complete algorithm implemented by the Timeout-based Power Manager is described by the pseudo-code 2. Let us assume that a burst-download is ongoing. In this case, TPM is executing lines 4-15, and the mobile host utilizes the PSM. Specifically, TPM waits the beginning of a new idle time (line 5). Then, in lines 6-15 it implements the timeout-based policy to distinguish between interarrival times and User Think Times. Specifically, it monitors the length of the ongoing idle time (line 8) until until one of the following conditions occur, i.e.: i) the idle time is longer than t_{TO} (lines 9-10); or ii) until a new segment either is received, or becomes ready to be sent (lines 11-12). In the latter case the burst download continues. Hence, TPM keeps executing the loop defined by lines 4-15, i.e., it waits for the next idle time. In the former case (i.e, when a User Think Time is detected), TPM switches the mobile host in the Off Mode (line 16). Then, it lets the mobile host in the Off Mode until a new burst download is detected, i.e., until a new request is generated by the application at the mobile host (lines 17-19). At this point in time, TPM switches the mobile host to the PSM algorithm (line 2), and starts monitoring the next idle times as explained above (lines 4-15).

7.6.3. Evaluating the Cross-layer Power Manager

In this Section we exploit the analytical model of Section 7.4 to evaluate the improvements in terms of power saving achieved by the Cross-layer Power Manager with respect to the standard 802.11 PSM. With regard to the Timeout-based Power Manager, we specialize its definition by assuming the simplest policy. Specifically, we only use the (sampled) average value of the Round Trip Time (RTT), and we define t_{TO} as $2 \cdot RTT$. The assumption behind this choice is that the probability of sampling a Round Trip Time more than twice longer than the average value is negligible. The performance of the Timeout-based Power Manager in four different cases is evaluated, corresponding to four different values of the average Round Trip Time, i.e. RTT=100 ms, 200 ms,

```
while true do
  switch to 802.11 PSM
  $is\_UTT = false$
  repeat
    wait the next idle time
    $it\_end = false$
    repeat
      $t =<$ update the idle-time length $>$
      if $t \geq t_{TO}$ then
        $is\_UTT = true$
      else if a new packet is ready then
        $it\_end = true$
      end if
    until $it\_end == true$ or $is\_UTT == true$
  until $is\_UTT == true$
  switch to Off Mode
  repeat
    <wait>
  until a new request is generated by the application
end while
```

Pseudo-code 2: Timeout-based Power Manager

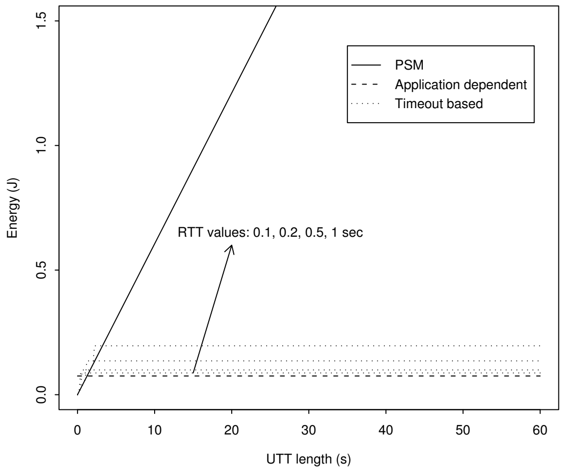


Figure 7.16.: Energy expenditure of Cross-layer Power Manager as a function of the User Think Time length.

500 ms and 1 s.

A first set of results is devoted to evaluate the sensitiveness of the Cross-layer Power Manager to the User Think Times. Figure 7.16 shows the energy consumption of the Timeout-based Power Manager for the different RTT values. The energy expenditure of the Application-dependent Power Manager and of the standard PSM are also shown. The energy consumption of PSM is computed based on Equation 7.22. On the other hand, the energy consumption of the Application-dependent Power Manager (E_{APM}) is constant, and is equal to $t_{oa} \cdot P_{ac}$ (recall that this Power Manager switches the mobile off immediately, and resumes it at the end of the User Think Time). Finally, the energy expenditure of the Timeout-based Power Manager is computed based on the following equation:

$$E_{TPM}\left(UTT\right) = \begin{cases} E_P\left(UTT\right) & UTT \leq t_{TO} \\ E_P\left(t_{TO}\right) + t_{oa} \cdot P_{ac} & UTT > t_{TO} \end{cases}, \qquad (7.23)$$

where $E_P\left(\cdot\right)$ is the energy expenditure of the standard PSM, defined by Equation 7.22. In detail, the Timeout-based Power Manager lets the mobile host in the standard PSM for User Think Times less than t_{TO}. Thus, in these cases the energy consumption is exactly the same achieved by the standard PSM, i.e., $E\left[t_{TO}\right]$. On the other hand, for User Think Times greater than t_{TO}, the Timeout-based Power Manager utilizes the standard PSM for the first t_{TO} seconds, and then it switches the mobile host off. Hence, the energy consumption is the energy spent in the first t_{TO} seconds (i.e., $E_P\left(t_{TO}\right)$), plus the energy spent to switch the mobile host to the active mode once the User Think Time is elapsed (i.e., $t_{oa} \cdot P_{ac}$). As highlighted in Section 7.6.2, the Application-dependent Power Manager performs worse than PSM for small User Think Time values. The same behavior is also exhibited by the Timeout-based Power Manager for User Think Times around the timeout value. However, the region where the Cross-layer Power Manager performs worse than PSM is limited to very small User Think Times, in the order of few seconds. As highlighted in Section 7.2, User Think Times are unlikely to be so short, since they are typically in the order of tens of seconds. Figure 7.16 shows that, for typical UTT values, the Cross-layer Power Manager significantly outperform PSM. Another interesting characteristic of Figure 7.16 is the comparison between the Timeout-based and the Application-dependent Power Manager. The Application-dependent Power Manager performs the best. The Timeout-based Power Manager consumes from 0.16 times to 2.6 times the energy spent by the Application-dependent Power Manager, for RTT equal to 0.1 s, and 1 s, respectively. However, the improvement with respect to PSM still remains very high. Moreover, it should be pointed out that the Timeout-based Power Manager can be implemented with very little cost at the mobile host either at the transport level or at the middleware level.

To complete the analysis of the Cross-layer Power Manager we now consider the energy spent not only during User Think Times, but also during burst-download phases. Specifically, we assume that during burst-download phases the Timeout-based Power Manager does not detect false User Think Times, i.e., we assume that $p\left(t_i \text{ is a UTT} | t_i \geq 2 \cdot RTT\right) \approx 1$ holds. Under this hypothesis, the Cross-layer Power Manager behaves exactly as PSM during burst-download phases. By exploiting the analytical formulations of E_{PSM}, E_{APM} and E_{TPM}, we have evaluated the energy spent by the standard PSM, by the Application-dependent Power Manager, and by the Timout-based Power Manager during the download of a single burst followed by a User Think Time. These quantities

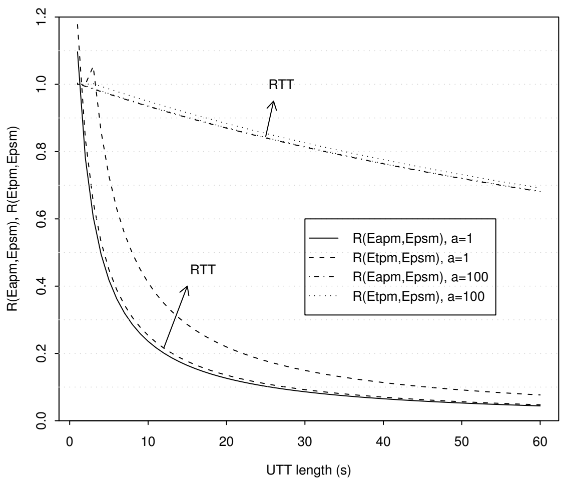


Figure 7.17.: Energy saving achieved by the Cross-layer Power Manager.

are throughout referred to as $E^{(1)}_{PSM}$, $E^{(1)}_{APM}$ and $E^{(1)}_{TPM}$, respectively. Then, we have evaluated the indexes $R(E^{(1)}_{APM}, E^{(1)}_{PSM})$ and $R(E^{(1)}_{TPM}, E^{(1)}_{PSM})$. Figure 7.17 plots these indexes as functions of the User Think Time length. With respect to the latter index, two curves are drawn for RTT equal to 0.1 s and 1 s, respectively. Furthermore, to characterize the CPM performance for a wide range of burst sizes, the curves have been replicated by setting the burst-size scaling factor (i.e., a) to 1 and to 100. For typical User Think Time values, the improvement over the standard 802.11 PSM is evident. For example, for small burst sizes (i.e., a=1), and User Think Time length equal to 30 s, the Application-dependent Power Manager spends around 8.6% of the energy spent by the PSM, while the Timeout-based Power Manager spends always less than 15% of the PSM energy. These values drop to 4.4% and 7.7%, respectively, when the User Think Time length is 60 s. As expected, the performance gains are reduced if we focus on a particular User Think Time length, and increase the burst sizes (i.e., set a to 100). This is because, for a given User Think Time length, the relative cost of downloading a burst becomes higher when the burst size increases (see Figure 7.14). Moreover, in this case the performance difference between the Application-dependent and the Timeout-based Power Managers is less marked. In detail, for User Think Times equal to 30 s and 60 s, the energy spent by the Cross-layer Power Manager is around 80% and 70% of the energy spent by PSM, respectively.

To summarize, the Cross-layer Power Manager shows significant power saving with respect to PSM. For typical values of the User Think Time (i.e., 30 s), the saving is in the order of 20% for large burst sizes, and becomes as high as 90% for small burst sizes.

7.6.4. Discussion

The results presented in Section 7.6.3 show that very simple mechanisms implemented beside the standard PSM can lead to significant improvements from a power-saving point of view. Our opinion is that these improvements stem from the cross-layer nature of the Cross-layer Power Manager design. Specifically, PSM just utilizes MAC-level information (i.e., availability of frames to/from the mobile host) to manage the mobile host's wireless interface. We have shown that this policy is not enough flexible to cope with the typical network traffic generated in Wi-Fi hotspots. On the other hand, the Cross-layer Power Manager dynamically chooses the best power-management policy based on simple information residing at the application or at the transport/middleware layers. The performance improvements presented so far show how such a cross-layer design is powerful.

Moreover, we would like to highlight the simplicity of the proposed improvements. From an architectural point of view, CPM can be entirely implemented at the mobile host. It does not require modifications to PSM, and hence can operate with commercial wireless cards. In a real case, where the hotspot is operated by some Internet Service Provider, the Cross-layer Power Manager could be a simple software module, which is shipped by the Internet Service Provider to its subscribers. Last but not least, possible non-subscriber users of a public hotspot will use PSM without noticing any performance degradation due to the presence of other "improved" users.

In the definition of the Cross-layer Power Manager we have assumed that i) a single network application is running at the mobile host; ii) this application uses a single TCP connection with the server; and iii) this application acts as a client, i.e., new bursts start with a request sent from the mobile host to the (fixed) server. As far as the first assumption, it should be noted that users within a Wi-Fi hotspot are very likely to run a single network application at once. This assumption relies on the observation that Web is ever more the killer Internet application [15]. Furthermore, many network applications that were originally designed without any relationship with Web, are nowadays accessed through Web technologies (e.g., e-mail, file download). Therefore, it is reasonable assuming that the "typical" hotspot user will use just the Web as network application.

The assumption of having a single TCP connection between the client and the server may fail, as popular Web browsers use parallel connections to download Web pages. However, as far as the Application-dependent Power Manager, it should be noted that the same algorithm can be used also in the case of multiple TCP connections. Specifically, APM detects User Think Times and new bursts irrespective of the number of TCP connection used. As far as the Timeout-based Power Manager, the definition should be slightly modified to support parallel connections. For example, TPM could monitor the traffic exchanged between the mobile host and the Access Point, irrespective of the particular transport-level connections, to detect User Think Times. A User Think Time would be detected when the applications at the mobile and fixed hosts do not exchange any data for t_{TO} seconds.

Finally, the Cross-layer Power Manager 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). Specifically, it is sufficient that, during User Think Times, CPM periodically switches the mobile host to the standard PSM. That way, frames that could have been buffered at the Access Point would be downloaded by exploiting the PSM mechanisms. Furthermore, CPM could switch again the mobile host to the Off Mode if no new data are exchanged for a Beacon Interval. This CPM extension will have some additional cost, since more switching-on events are required, and more time would be spent in PSM. With respect to this scenario, the results we have presented here represent an upper bound to the power saving achieved by CPM.

8. Conclusions and Future Works

In this work we have faced the problem of reducing the power consumption of mobile hosts accessing the Internet through wireless LANs (i.e., within wireless hotspots). This is a very pressing issue, since the increase of battery technology is not able to fullfil the mobile hosts' requirements. Our work focuses on defining power-saving architectures and protocols for reducing the power consumption of the mobile hosts' networking subsystem. Past experimental measurements have shown that networking is responsible for up to 50% of the global device energy consumption. Hence, it is vital to design power-efficient networking solutions.

We have focused on a typical Wi-Fi hotspot scenario, where a user accesses the Internet by means of a wireless mobile host. We have considered best-effort type of applications, such as Web, e-mail, file download, which are the most popular applications in the today Internet. It is well-known that the major source of power wastage in this scenario are *idle times* in the network traffic, i.e., time intervals during which the wireless interface of the mobile host is power on without exchanging any data. We have highlighted that idle times in best-effort traffic can be classified in interarrival times (i.e., short idle times inside bursts of packets exchanged by the mobile device), and User Think Times (i.e., long idle times between consecutive bursts). Due to their different nature, interarrival times and User Think Times have very different lengths. Interarrival times are typically below 1 s, while User Think Times can be as long as 60 s and beyond. The best technique to deal with this problem is turning the wireless interface in a low-power operating mode (possibly, switching it off) during these inactivity phases.

We have firstly explored power-saving solutions operating exclusively at the middleware layer. The advantage of this approach is that it is independent on the particular wireless technology. Hence such solutions are highly portable, and can operate also in heterogeneous environments. Furthermore, by operating at the middleware layer, they can exploit clear knowledge about the application behavior. We have explored application-dependent and application-independent policies. Application-dependent policies exploit *a-priori* information about the application-level traffic profile. Based on these information, and by monitoring the traffic on-line, they decide when and for how long the wireless interface should be kept off, because no data are expected to flow on the network. On the other hand, application-independent policies just rely on monitoring on-line the application-level traffic profile. They build statistical models of the traffic, and exploit the models to predict future inactivity periods (i.e., time intervals during which no data are expected to flow on the network). The wireless interface is switched off for the (predicted) duration of these inactivity periods. Both policies rely on the standard Indirect-TCP model, and exploit the Indirect-TCP Daemon (a sw agent running at the Access Point), to buffer possible packets addressed to the mobile host while it is disconnected.

We have designed two power-saving systems, named PS-Web and PS-WiFi, which follow the application-dependent and independent paradigm, respectively. We have evaluated both of these solutions in depth. Specifically, we have developed a real-Internet prototype implementing both PS-Web and PS-WiFi, and we have run an extensive set of measurements. PS-Web has been tested with respect to Web, which produces the lion's share of the traffic in the today Internet. With respect to a standard I-TCP architecture, PS-Web is able to save around 90% of the energy, with negligible degradation in the QoS perceived by the user. Specifically, it introduces – on average – 200 ms to the donwload of Web pages. Clearly, PS-WiFi performs worse than PS-Web. However, it is still able to save around 80% of the energy spent in a traditional I-TCP architecture, by introducing just 400 ms (on averge) to the Web-page download time. PS-WiFi has been also analyzed in the case of e-mail traffic, and in the case of mixed traffic (i.e., Web and e-mail). Experimental results show that also in this cases PS-WiFi is highly efficient. Finally, we have developed an analytical model of its behavior, and we have assessed the PS-WiFi sensitiveness with respect to key Internet parameters, i.e., the throughput on the wireless and wired parts of the network, and the Internet Round Trip Time. We have found that PS-WiFi is very efficient for a broad range of these parameters.

Based on these results, we have concluded that PS-Web is best suited for dedicated environments, where the set of applications is known at design time. On the other hand, PS-WiFi is a better choice in general-purpose environments, where no assumptions about the applications can be done (PS-WiFi just requires that applications do not have real-time requirements).

One of the advantages of PS-Web and PS-WiFi is that they can operate irrespective of the wireless technology. From a different standpoint, this advantage may turn into a weakness. Indeed, for the sake of portability, PS-Web and PS-WiFi can just switch off the wireless interface of the mobile host, but cannot exploit low-power operating modes that are typically defined by specific wireless technologies. We have highlighted that, due to the relative long time needed to switch on a wireless interface, PS-Web and PS-WiFi can obtain sub-optimal power-saving during interarrival times.

Therefore, we have analyzed the power-saving performance of the de-facto standard technology, i.e., IEEE 802.11. The standard 802.11 defines a Power-Saving Mode (PSM) that exploit the sleep mode of the wireless interface to conserve energy. In the sleep mode, just a small portion of the interface circuitry is powered on. The performance of PSM has been evaluated through analytical and simulation models. Firstly, we have assessed PSM ability to manage interarrival times, i.e., we have evaluated its behavior during bursts. The power saving achieved by using PSM can be as high as 90%. We have also highlighted the dependence of PSM performance on the main parameters that characterize the network traffic during bursts, i.e., the average bursts' size, and the transport-level throughput. The energy saving achieved by PSM is almost independent on the average bursts' size. The transport-level throughput is mainly defined by the packet-loss probability (p_l^{tcp}) and the Round Trip Time (RTT). When either p_l^{tcp} or RTT increases, the power consumption increases, whether PSM is active or not. However, PSM significantly mitigates the additional power consumption. Similar observations have been derived with respect to the WLAN congestion (i.e., the number of users within the same hotspot). The power consumption increases with the congestion level, due to TCP- and MAC-level reasons. PSM mitigates the effects of the former, while it is not effective against the latter. We have sketched a MAC-level modification that can cope with this problem. By means of analytical results, we have shown that PSM achieves near-optimal performance in contrasting energy wastages related to interarrival times. Then, we have evaluated PSM ability to manage User Think Times. In this case PSM has shown to be far from optimal. We have highlighted that this is due to using the sleep operating mode. During User Think Times, switching the wireless interface off is much more energy efficient.

All in all, the power-saving performance of PSM is comparable with that of PS-Web and PS-WiFi. However, in some way such approaches are complementary. PSM is very effective during interarrival times, while PS-Web and PS-WiFi perform the best during User Think Times.

We have thus proposed a Cross-layer Power Manager (CPM), that uses PSM during bursts, and switches the wireless interface off during User Think Times. The main issue in the CPM definition is how to detect the beginning of User Think Times and bursts. To this end, CPM includes mechanisms originally designed for PS-Web and PS-WiFi. Specifically, we have proposed an implementation of the Cross-layer Power Manager, named Application-dependent Power Manager, that exploits information about the application behavior. Moreover, we have proposed an implementation of the Cross-layer Power Manager, named Timeout-based Power Manager, that is application independent, and relies on timeouts to perform detections. The design of CPM is cross-layer in nature, since it exploit information residing at the MAC-, middleware- and application-level to conserve energy.

We have evaluated CPM in depth. Specifically, For a broad range of User Think Time values and burst sizes the Cross-layer Power Manager has shown to achieve power saving between 20% and 90% with respect to the standard 802.11 PSM. This result shows how much a cross-layer design outperforms policies operating just at a single level of the protocol stack.

The main contributions of this work are as follows. We have provided a deep characterization of the 802.11 Power-Saving Mode. We have defined and evaluated power-saving policies operating at the middleware layer. We have provided an analytical framework that highlights which policy is best suited, depending on the type of idle times present in the application-level traffic. Then, we have shown how a Cross-layer design can integrate and improve the performance of MAC- and middleware-level policies in isolation. Finally, we have proposed two implementations of the Cross-layer Power Manager that do not require hardware modifications, and hence can be included with little effort in current 802.11 wireless hotspots.

Bibliography

[1] S. Agrawal and S. Singh, "An Experimental Study of TCP's Energy Consumption over a Wireless Link", 4th European Personal Mobile Communications Conference, February 20-22, 2001, Vienna, Austria. 36

[2] M. Anand, E. Nightingale, and J. Flinn, "Self-Tuning Wireless Network Power Management", in Proc. of the Ninth ACM Annual International Conference on Mobile Computing and Networking (MobiCom 2003), San Diego (CA), Sept. 14-19, 2003. 22

[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. 110, 113

[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. 87

[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, IEEE Computer Society Press.

[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. 70, 74, 113

[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. 14, 36, 74

[8] G. Anastasi, M. Conti, E. Gregori and A. Passarella, "Power-Saving in Wi-Fi Hotspots: an Analytical Study", Proceedings of the Eighth International Conference on Personal Wireless Communications (PWC 2003) - LNCS Series, Venice (I), September 23-25, 2003.

[9] G. Anastasi, M. Conti, E. Gregori and A. Passarella, "Balancing Energy Saving and QoS in the Mobile Internet: An Application-Independent Approach", Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS-36), Hawaii, January 6-9, 2003.

[10] G. Anastasi, M. Conti, E. Gregori and A. Passarella, "A Power Saving Architecture for Web Access from Mobile Computers", Proceedings of the 2nd IFIP-TC6 NETWORKING Conference (Networking 2002) - LNCS Series, Pisa (I), May 19-24, 2002.

[11] Apache Web Server on-line documentation, available at `http://www.apache.org`. 68

[12] M. Arlitt and C. Williamson, “Internet Web Servers: Workload Characterization and Performance Implication”, IEEE/ACM Transactions on Networking, Vol. 5, No. 5, pp. 631-645, October 1997. 30

[13] 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. 30, 100, 101

[14] V. Baiamonte and C.-F. Chiasserini, “An Energy-efficient MAC layer Scheme for 802.11-based WLANs", Proc. of EWCN, 2004. 22

[15] 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`. 14, 83, 118

[16] 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. 27, 29, 34, 37, 40, 104

[17] 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, December 1997. 27, 85, 105

[18] P. Barford, A. Bestavros, A. Bradley and M. Crovella, “Changes in Web client access patterns: Characteristics and caching implications”, World Wide Web (Special Issue on Characterization and Performance Evaluation), 1999. 30, 66, 67

[19] 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. 30, 32, 39, 41, 61, 66, 67, 86, 98, 100, 101

[20] L. Bertolotti, M. C. Calzarossa, “Workload Characterization of Mail Servers”, Proc. SPECTS 2000, Vancouver, (Canada), July 2000. 63

[21] 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. 23, 107

[22] 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. 86, 141

[23] 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. 23

[24] M. Conti, G. Maselli, G. Turi, S. Giordano, “Cross-Layering in Mobile Ad Hoc Network Design”, IEEE Computer, Vol. 37, N. 2, February 2004, pp. 48-51. 112

[25] 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. 30, 32, 41, 86, 100, 101

[26] C. Cunha, A. Bestavros and M. Crovella, “Characteristics of WWW Client-Based Traces”, Technical Report TR-95-010, Boston University Department of Computer Science, April 1995. 30, 41

[27] F. Donelson Smith, F. Herndndez Campos, K. Jeffay and D. Ott, “What TCP/IP Protocol Headers Can Tell Ue About the Web”, ACM SIGMETRICS 2001, pp. 245-256. 85, 100, 101

[28] 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

[29] L.M. Feeney and M. Nilsson, “Investigating the energy consumption of a wireless network interface in an ad hoc networking environment”, Proc. of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001), Anchorage (Alaska) Apr. 22-26, 2001. 86

[30] J. Flinna and M. Satyanarayanan, “Managing Battery Lifetime with Energy-Aware Adaptation”, ACM Transactions on Computer Systems, Vol. 22, No. 2, May 2004, pp. 137–179. 23, 24

[31] 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. 23, 24

[32] D.P. Helmbold, D.E. Long and B. Sherrod, “A Dynamic Disk Spin-down Technique for Mobile Computing”, Proc. of the 2nd Annual ACM International Conference on Mobile Computing and Networking (MobiCom ’96), pp. 130-142, Nov. 1996. 56, 110

[33] M. Herbster and M.K. Warmuth, “Tracking the best expert”, Proc. of the 12th International Conference on Machine Learning, pp. 286-294, 1995. 53, 55, 56

[34] R. Fielding, J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee, “Hypertext Transfer Protocol - HTTP/1.1”, RFC 2068, 1997. 68, 85

[35] IEEE standard for Wireless LAN- Medium Access Control and Physical Layer Specification, 802.11, November 1997. 83, 86, 93, 94, 95, 143

[36] 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. 92

[37] A. Joshi, “On proxy agents, mobility, and web access”, ACM/Baltzer Mobile Networks and Applications, Vol. 5 (2000), pp. 233-241. 23

[38] R. Krashinsky and H. Balakrishnan, “Minimizing Energy for Wireless Web Access with Bounded Slowdown”, Proceedings of Proceedings of the Eighth Annual ACME/IEEE International Conference on Mobile Computing and Networking (MOBICOM’02), 2002. 21, 22, 86

[39] 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. 14, 22, 23, 29

[40] A. M. Law and D. Kelton, “Simulation Modeling and Analysis” (Second Edition), McGraw-Hill, 1991. 45

[41] J. Lee, C. Rosenberg, and E.K.P. Chong, “Energy Efficient Scheduler Design in Wireless Networks”, Proceedings of the Second Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt 2004), Cambridge (UK), March 24-26, 2004. 22, 23

[42] 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. 23

[43] M. Mathis, J. Semke, J. Mahdavi and T.Ott, “The macroscopic behavior of the TCP Congestion Avoidance Algorithm”, Computer Communication Review, Vol. 27, No. 3, July 1997. 68

[44] S. Nath, Z. Anderson and S. Seshan, “Choosing Beacon Periods to Improve Response Times for Wireless HTTP Clients", Proc. of ACM MobiWac’04, Oct. 1, 2004. 21, 22

[45] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, “Modeling TCP Throughput: A Simple Model and its Empirical Validation”, in ACM Sigcomm, 1998. 101, 103

[46] A. Passarella, “Un’architettura power-saving per l’accesso al Web da computer mobile”, Laurea Thesis, University of Pisa (I), Oct. 2001. 35, 47

[47] S.H. Phatak, V. Esakki, B.R. Badrinath and L. Iftode, “Web&: An Architecture for Non-Interactive Web”, Proc. of the 2nd IEEE Workshop on Internet Applications (WIAPP01), S. Jose (CA), July 23-24, 2001. 23

[48] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt and P. Sinha, “Understanding TCP fairness over Wireless LAN”, Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003), S. Francisco (CA), Mar. 30 - Apr. 3, 2003. 68

[49] N. Ramos, D. Panigrahi and S. Dey, “Energy-Efficient Link Adaptations in IEEE 802.11b Wireless LAN", Proc. of WOC, 2003. 22

[50] 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. 23

[51] 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. 14

[52] 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. 22, 23, 29

[53] W.R. Stevens, “TCP/IP Illustrated, Volume 1: The Protocols” , Addison-Wesley, 1994. 40, 85

[54] H. Yan, R. Krishnan, S.A. Watterson and D.K. Lowenthal, “Client-Centered Energy Savings for Concurrent HTTP Connections", Proc. of NOSSDAV’04, June 16-18, 2004. 22, 23

[55] Wireless World Research Forum (WWRF): `http://www.ist-wsi.org`. 13

Part IV.

Appendices

A. Energy Consumption of the PS-Web System

This Appendix contains the proof of the Proposition 1. In a system that adopts the global strategy, the energy consumption is:

$$C_{global} = \sum_{i=1}^{N} \left(\frac{D_i + \beta_i}{\gamma} \right) + A + (s+1) \cdot t_{so} \ . \tag{A.1}$$

Equation 5.10 is derived from Equation 5.5 by setting $\gamma_i = \gamma$, $\tau_i = 0$, $U = 0$ and $m = s + 1$. Herafter we prove the above claims.

Firstly, by the definition of global strategy, $U = 0$.

Moreover, in the global strategy, the main file and the embedded files are transferred over the wireless link when they are already stored at the Access Point. Therefore, the wireless link is used at its full available throughput, and hence γ_i in Equation 5.5 becomes γ.

τ_i is negligible. This result derives from the following considerations: i) the PS-Daemon includes Web proxy functionalities; ii) the RTT between the mobile host and the Access Point is typically negligible; and iii) the overhead related to pre-fetching can be included in β_i, A and s as shown below.

The pre-fetching mechanism forces the mobile host to request a number of residual-transfer-time estimates from the PS-Daemon during the Active Phase. Let g be this number ($g > 0$). When one of these estimates is greater than t_{so} the mobile host switches the wireless interface off. If s is the number of estimates greater than t_{so} then $m = s + 1$. Moreover, the time intervals during which the wireless interface remains on during idle periods within the Active Phase correspond to the estimates less than t_{so}. Therefore, $A \leq (g - s) \cdot t_{so}$. It must be pointed out that g is a random variable. Its distribution is very complex and depends on: i) the residual-transfer-time estimation algorithm; ii) the throughput between the PS-Daemon and the server; iii) the number and size of embedded files. Therefore, a closed formula for g is almost impossible to derive, and for this reason, to study the effectiveness of the global strategy, we performed an experimental analysis (see Sections 5.6 and 5.7).

Finally, β_i is made up of two components, and can be expressed as $\beta_i = h_i + p_i$, where h_i is the same as in Equation 5.2, while p_i is the overhead introduced by the residual-transfer-time estimation process associated with the i-th file of a Web page. Specifically, during the Active Phase, the mobile host exchanges messages with the PS-Daemon to receive residual-transfer-time estimates. The size of these messages is nearly constant, and can be approximated by the average size, $\bar{q}$. Therefore, $\bar{q} \cdot g$ is the overhead (in bytes) of a Web-page download, and $p_i = \bar{q} \cdot g / N$ is the overhead related to the i-th file transfer.

This concludes the proof.

B. Energy Consumption of the PS-WiFi System

This Appendix contains the proof of Theorem 2.

Theorem 2: *The energy spent to download a single basic block by using PS-WiFi is*

$$C_{ps} = \frac{B}{\overline{\gamma}_{wl}} + t_{so} \cdot \left\{ \frac{B}{\overline{\gamma} \cdot \overline{RTT}} \cdot S_1 + l \cdot \left(F + \lceil \log_2 \overline{UTT} \rceil\right) + p\left(u^{(0)} > t_{so}\right) \right\} \ , \tag{B.1}$$

where i) S_1 is the average number of switching-on events occurring during a short idle time; ii) F is the number of switching-on events occurring during a long idle time, before the long idle-time estimator is invoked (i.e., after $u^{(2)}$ of Equation 6.1 is generated); and iii) $p\left(u^{(0)} > t_{so}\right)$ is the probability of $u^{(0)}$ being greater than t_{so}.

Proof. Based on Equation 6.3, to characterize C_{ps} we have to derive a closed form for the average number of switching-on events occurring during a basic-block download, i.e., S. S can be expressed as in the following proposition:

Proposition 10 *The average number of switching-on events during the download of the basic block is*

$$S = r \cdot S_1 + l \cdot S_2 + S_3 \ , \tag{B.2}$$

where: i) r and l are the number of short and long idle times during the basic-block download, respectively; ii) S_1 and S_2 are the number of switching-on events during a short and a long idle time, respectively; and iii) S_3 is the number of switching-on events occurring during the download of the first Web page, after the main file has arrived at the mobile device, and before the HTTP Request(s) for the embedded files have been sent.

Proof. As discussed in Section 6.4.2, PS-WiFi regenerates with respect to the point in time when an idle time occurs. Therefore, S can be derived as follows: i) we evaluate the number of switching-on events related to a generic short idle time, and related to a generic long idle time (i.e., S_1 and S_2); ii) we evaluate the number of short idle times and long idle times occurring during the basic-block download (i.e., r and l); and iii) due to the regenerative property, we derive the total number of switching-on events as $r \cdot S_1 + l \cdot S_2$. Finally, it must be noted that a further idle time is detected by PS-WiFi when the main file of the first page has been downloaded. In fact, in this case there are no more data to be exchanged on the WLAN, and hence the estimator is invoked. Therefore, an additional switch-on event may occur, and the term S_3 accounts for this case. ■

To achieve a closed form of S we have to characterize the terms composing Equation B.2. The following lemmas are devoted to this task. For ease of reading, the distribution of the number of switching-on events occurring during a short-idle time is postponed to Appendix C. S_1 is the average value of this distribution.

Lemma 10 *The average number of short idle times occurring during the download of the basic block (i.e., r) is*

$$r = \frac{B}{\overline{\gamma} \cdot \overline{RTT}} \ . \tag{B.3}$$

Proof. As shown in Section 6.4.3.1, in our model the data transfers over the TCP connection between the I-TCP Daemon and the Web server occur as follows: i) once every RTT the TCP at the Web server sends a fixed number of back-to-back TCP segments; and ii) these back-to-back TCP segments arrive at the Access Point together. Therefore, the average number of bytes transferred during each RTT is $\beta \triangleq \overline{\gamma} \cdot \overline{RTT}$. Moreover, since r can be seen as the number of $RTTs$ within a basic block, r is equal to $B\,/\beta$. ■

Lemma 11 *The average number of switching-on events during a long idle time (i.e., S_2) is*

$$S_2 = F + \left\lceil \log_2 \overline{UTT} \right\rceil \ , \tag{B.4}$$

where F is the average number of switching-on events occurring before the backoff procedure starts.

Proof. By definition, User Think Times are greater than $1sec$. From the sequence $u^{(i)}$ shown in Equation 6.1, it appears that PS-WiFi needs 2 updates to start the exponential backoff procedure. This procedure starts when the idle time results greater than $1\,sec$, i.e., after the PS-WiFi has generated: i) $u^{(0)}$ as the first estimate; ii) $u^{(1)}$ (i.e., the 90$^{\text{th}}$percentile of short idle times) as the first update of the estimate; and iii) $u^{(2)}$ (i.e., 1 sec) as the second update. Therefore, at this point in time, up to 3 switching-on events have occurred. The analysis of the number of switching-on events that occur up to this point (i.e., F) is very similar to the analysis of the number of switching-on events that occur during a short idle time (i.e., S_1), derived in Appendix B. For the sake of space, we here omit this analysis. Further estimate updates are derived according the binary exponential backoff procedure, as shown in Equation 6.1. It is easy to show that, if Q denotes the length of a long idle time, the number of estimate updates generated by the backoff procedure is $\lceil \log_2 Q \rceil$. Moreover, as upon each such update the wireless interface of the mobile device is shut down (i.e., each value of the $z^{(i)}$ sequence is greater than 1 sec, and hence it can be reasonably assumed to be greater than t_{so}), $\lceil \log_2 Q \rceil$ is also the number of switching-on events that occur during the backoff procedure. The expression of S_2 can be derived immediately by recalling that, on average, Q is equal to $\overline{UTT}$. ■

Lemma 12 *The average number of switching-on events occurring after the download of the first main file of the basic block, before the request for the embedded files is:*

$$S_3 = p\left(u^{(0)} > t_{so}\right) \ . \tag{B.5}$$

Proof. When the Access Point sends the last packet of the main file, no more data are available to be exchanged on the WLAN, and hence an idle time is detected. Therefore, PS-WiFi generates an idle-time estimate (i.e., $u^{(0)}$), and the mobile device switches the wireless interface off if this estimate is greater than t_{so}. Furthermore, as the packet arrives at the application level, the browser sends the HTTP Request(s) for the embedded files. Thus, a single switching-on event may occur in this case, if the estimate provided by the PS-PT at the Access Point is sufficiently long. The average number of switching-on events in this case is hence the probability of $u^{(0)}$ being greater than t_{so}.

■

Finally, the closed form for C_{ps} claimed in Theorem 2 is derived by substituting results from the above lemmas in Equation 6.3. This concludes the proof. ■

C. Switching-on Events in the PS-WiFi System

In this Appendix the distribution of the number of switching-on events occurring during a short idle time is derived.

Lemma 13 *The distribution of the switching-on events occurring during a short idle time is as follows:*

$$\begin{aligned}
p\,(0\,\textit{switching-on event}) = \;& p\,(t' \leq t_{so},\, t' \geq t) + \\
+ \;& p\,(t' \leq t_{so},\, t' < t \leq k,\, k - t' \leq t_{so}) + \\
+ \;& p\,(t' \leq t_{so},\, k - t' \leq t_{so}) \cdot (1 - \chi\,(k,\, t_{so})) \\
p\,(1\,\textit{switching-on event}) = \;& p\,(t' > t,\, t' > t_{so}) + \\
+ \;& p\,(t' < t,\, t' \leq t_{so},\, k > t,\, k - t' > t_{so}) + \\
+ \;& p\,(t' < t,\, t' \leq t_{so}, \\
& k < t,\, k - t' \leq t_{so}) \cdot \chi\,(k,\, t_{so}) \\
p\,(2\,\textit{switching-on events}) = \;& p\,(t' < t,\, t' \leq t_{so}, \\
& k < t,\, k - t' > t_{so}) \cdot \chi\,(k,\, t_{so}) + \\
+ \;& p\,(t' < t,\, t' > t_{so},\, k > t,\, k - t' > t_{so}) + \\
+ \;& p\,(t' < t,\, t' > t_{so}, \\
& k < t,\, k - t' \leq t_{so}) \cdot \chi\,(k,\, t_{so}) \\
p\,(3\,\textit{switching-on events}) = \;& p\,(t' < t,\, t' > t_{so}, \\
& k < t,\, k - t' > t_{so}) \cdot \chi\,(k,\, t_{so})
\end{aligned} \tag{C.1}$$

where $\chi\,(k,\, t_{so})$ *is defined by Equation 6.9.*

Proof. When a short idle time begins, the Variable-Share Update algorithm provides an estimate, t', of the actual idle time, t. The wireless interface is switched off if t' is greater than t_{so}. If t' is less than t, the estimate is updated with the 90$^{\text{th}}$ percentile of the short idle times, i.e. k. The wireless interface of the mobile device is switched off again only if $k - t$ is greater than t_{so}. Finally, if t is even greater than k, the algorithm executes a binary exponential backoff procedure starting from 1 second, and hence the wireless interface is switched off if $1\,sec - k$ is greater than t_{so}. Therefore, since t is less than 1 sec by definition, there can't be more than 3 switching-on events within a short idle time. Moreover, a specific number of switching-on events is achieved in different cases, according to the relative values of t, t', k and t_{so}. Therefore, each term of the switching-on-event distribution must be computed as the sum of the marginal probabilities of each case. It is worth noting that, when t is greater than k, the wireless interface is switched off only if $1\,sec - k$ is greater than t_{so}. Since k and t_{so} are not random variables, we introduce $\chi\,(k,\, t_{so})$ in Equation C.1 to include this case. The single expressions provided in Equation C.1 can be easily derived from these remarks. ■

D. Additional Delay in the PS-WiFi System

This Appendix contains the proof of Theorem 4.

Theorem 4: *The average delay added by PS-WiFi to a packet flowing in the downlink direction is*

$$\begin{aligned} \overline{d} &= \tfrac{1}{2}\Big(\tfrac{M^2-t_{so}^2}{4M}\cdot u\left(M,\,t_{so}\right)+0.9\cdot\tfrac{k^2-t_{so}^2}{4M}\cdot u\left(k,\,t_{so}\right)+ \\ &+ 0.1\cdot\tfrac{2sec-M-k}{2}\cdot\chi\left(k,\,t_{so}\right)\Big) \end{aligned} \quad , \tag{D.1}$$

where $u\left(x,\,y\right)$ and $\chi\left(k,\,t_{so}\right)$ are defined as follows:

$$u\left(x,\,y\right)=\begin{cases}1 & \textit{if}\, x\geq y\\ 0 & \textit{otherwise}\end{cases} \quad , \quad \chi\left(k,\,t_{so}\right)=\begin{cases}1 & \textit{if}\ 1sec-k>t_{so}\\ 0 & \textit{otherwise}\end{cases} \quad . \tag{D.2}$$

Proof. The starting point of the proof is Proposition 5, that has been discussed in Section 6.4.3.1.
Proposition 5: *The average delay added by PS-WiFi to a packet flowing in the downlink direction can be expressed as*

$$\begin{aligned} \overline{d}=E\left[d\right]= &\ E\left[d\,|t'>t\right]\cdot p\left(t'>t\right)+E\left[d\,|t'<t\right]\cdot p\left(t'<t\right)= \\ = &\ \tfrac{1}{2}\cdot\left(E\left[d\,|t'>t\right]+E\left[d\,|t'<t\right]\right) \ . \end{aligned} \tag{D.3}$$

To obtain the closed form of Theorem 4, we have to separately analyze the two components highlighted in Proposition 5, i.e., i) the average delay when the initial short idle-time estimate is too large (i.e., when $t'>t$); and ii) the average delay when the initial short idle-time estimate is too short (i.e., when $t'<t$). The following lemmas provide closed forms for these components.
Lemma 1: *The average delay when t' is greater than t is*

$$E\left[d\,|t'>t\right]=\frac{M^2-t_{so}^2}{4M} \ . \tag{D.4}$$

Proof. When t' is greater than t, an additional delay is added only if t' is greater than t_{so}. Otherwise, the wireless interface of the mobile device remains on, and the new packet is received without being delayed. If t' is greater than t, the delay is $t'-t$. Therefore, the following equation holds:

$$E\left[d\,|t'>t\right] \ = \ E\left[t'-t\,|t'>t,\,t'>t_{so}\right]\cdot p\left(t'>t_{so}\right) \ . \tag{D.5}$$

The second term of equation D.5 (i.e., $p\,(t' > t_{so})$) can be computed from the t' distribution law:

$$p\,(t' > t_{so}) = \frac{M - t_{so}}{M} \quad . \tag{D.6}$$

Furthermore, the first term of Equation D.5 can be evaluated as follows:

$$\begin{array}{rl} E\,[t' - t\,|t' > t,\, t' > t_{so}] = & \int_{t_{so}}^{M} E\,[t' - t\,|t' > t,\, t' > t_{so}\;,\, t'] \cdot p\,(t')\, dt' \\ = & \frac{M+t_{so}}{4} \end{array} \quad , \tag{D.7}$$

where the closed formula for the integral is obtained by some algebraic manipulations. By substituting Equations D.6 and D.7 in Equation D.5 we obtain the closed form of Lemma *1*. ■

Lemma 2: *The average delay when t' is less than t is*

$$E\,[d\,|t' < t] = 0.9 \cdot \frac{k^2 - t_{so}^2}{4M} + 0.1 \cdot \frac{2sec - M - k}{2} \cdot \chi\,(k,\, t_{so}) \quad . \tag{D.8}$$

Proof. When t' is less than t, t' is updated by using the 90th percentile of the short idle-time distribution, i.e., k. In this case $\overline{d}$ can be evaluated by considering two possible cases in isolation, i.e., i) $t \leq k$, and ii) $t > k$. If t is less than k, the additional delay is $k - t$ seconds. More precisely, this delay is introduced only if the wireless interface is switched off after updating t', i.e., only if $k - t'$ is greater than t_{so} (see Equation 6.1). Therefore, the probability that the delay is $k - t$ can be expressed as the joint probability of the events $t \leq k$ and $k - t' > t_{so}$. Moreover, since t and t' are independent, the joint probability $p\,(t \leq k,\, k - t' > t_{so})$ can be computed as the product of the marginal probabilities of the two events. The same line of reasoning can also be followed to evaluate the average delay when t is greater than k. Specifically, the delay is $1\,sec - t$ if $1\,sec - k$ is greater than t_{so}, while it is 0 otherwise. To include this condition into our model, we use $\chi\,(k,\, t_{so})$, as defined in Equation 6.9. Therefore, $E\,[d\,|t' < t]$ can be expressed as follows:

$$\begin{array}{rl} E\,[d\,|t' < t] = & E\,[k - t\,|t' < t,\, t \leq k,\, k - t' > t_{so}] \cdot p\,(t \leq k) \cdot p\,(k - t' > t_{so}) + \\ & + E\,[1sec - t\,|t' < t,\, t > k] \cdot p\,(t > k) \cdot \chi\,(k,\, t_{so}) \quad . \end{array} \tag{D.9}$$

We are now in the position to derive all the components of Equation D.9. Firstly, the terms related to distribution law of t and t' can be easily computed:

$$\begin{array}{l} p\,(t \leq k) = 0.9 \\ p\,(t > k) = 0.1 \\ p\,(k - t' > t_{so}) = p\,(t' < k - t_{so}) = \frac{k - t_{so}}{M} \end{array} \quad . \tag{D.10}$$

Furthermore, by following the same line of reasoning used to derive (D.7), we can compute the component[1] $E\left[k-t\,|t'<t,\,t\le k,\,k-t'>t_{so}\right]$ of Equation D.9:

$$\begin{array}{rcl} E\left[k-t\right] & = & \int_0^{k-t_{so}} p\left(t'\right)\cdot\left(k-E\left[t\right]\right)dt' = \\ & = & \int_0^{k-t_{so}} p\left(t'\right)\cdot\left(k-\int_{t'}^{k} t\cdot p\left(t\right)dt\right)dt' = \\ & = & \frac{k+t_{so}}{4} \end{array} \quad . \tag{D.11}$$

The steps highlighted in Equation D.11 are derived as follows: i) the formula on the first line is obtained by fixing t' and integrating on its possible values; ii) the formula on the second line is obtained by using the average value definition to expand $E\left[t\right]$; and iii) the closed formula on the third line is obtained after simple computations.
The last step to evaluate Equation D.9 is the computation of $E\left[1sec-t\,|t'<t,\,t>k\right]$. By applying the same technique used to derive Equations D.7 and D.11, we obtain the following form[2]:

$$\begin{array}{rcl} E\left[1sec-t\right] & = & \int_k^{M} p\left(t\right)\cdot\left(1sec-t\right)dt \\ & = & 1sec-\frac{M+k}{2} \end{array} \quad . \tag{D.12}$$

Finally, by substituting Equations D.10, D.11 and D.12 in Equation D.9, we obtain a closed formula for $E\left[d\,|t'<t\right]$:

$$E\left[d\,|t'<t\right] = 0.9\cdot\frac{k^2-t_{so}^2}{4M} + 0.1\cdot\frac{2sec-M-k}{2}\cdot\chi\left(k,\,t_{so}\right) \quad . \tag{D.13}$$

■

Equations D.4 and D.13 allow us to complete our analysis by deriving a closed formula for the average value of d. Specifically, Equation D.3 becomes as follows:

$$\overline{d} = \frac{1}{2}\left(\frac{M^2-t_{so}^2}{4M} + 0.9\cdot\frac{k^2-t_{so}^2}{4M} + 0.1\cdot\frac{2sec-M-k}{2}\cdot\chi\left(k,\,t_{so}\right)\right) \quad . \tag{D.14}$$

Then, by recalling the definition of $\chi\left(k,\,t_{so}\right)$, Equation D.14 can be expressed as:

$$\overline{d}\equiv\overline{d}\left(M,\,t_{so}\right) = \begin{cases} 0.169\cdot M - 0.238\cdot\frac{t_{so}^2}{M} + 50msec & \text{if } 1\,sec-0.9\cdot M > t_{so} \\ 0.216\cdot M - 0.238\cdot\frac{t_{so}^2}{M} & \text{otherwise} \end{cases} \quad . \tag{D.15}$$

Finally, it must be noted that all the above equations rely on the assumption that $M>k\ge t_{so}$ holds. More generally, it is easy to show that $\overline{d}$ is as shown in Theorem 4. This concludes the proof of Theorem 4. ■

[1]For easy of reading, we omit of explicitly indicating the conditions who this average value obeys. However, they are explicitly shown in Equation D.9.
[2]Also in this case, we omit of explicitly indicating the conditions who this average value obeys.

E. MAC delay in the 802.11 PSM

In this appendix we derive a closed form expression of the average MAC delay experienced by the tagged mobile host, i.e., $E\left[tmac\right]$. Before proceeding, it is worth discussing some assumptions used to model the Wi-Fi hotspot.

E.1. Modeling the Wi-Fi hotspot

In our scenario, at any point in time, M background mobile hosts are active in the hotspot, i.e., they have a frame ready for transmission (see Figure E.1). We assume that background mobile hosts do not use the standard IEEE 802.11 protocol, but the p-persistent IEEE 802.11 protocol presented in [22]. The p-persistent protocol differs from the standard one in the way backoff intervals are selected. Specifically, after the channel is free for a DIFS interval, each mobile host having frames in the sending queue starts a transmission in the next slot with probability p, and defers to the following slot with probability $1 - p$. The value of p depends on the number of active mobile hosts and is chosen based on the following line of reasoning. When the standard 802.11 protocol is used, each mobile host experiences several backoff intervals before transmitting a frame successfully. If mobile hosts operate in asymptotic conditions, it is possible to derive an average value, say $E\left[Bk\right]$, for the length of these backoff intervals [22]. The value of p is chosen in such a way that the average backoff interval obtained by using the p-persistent protocol is equal to $E\left[Bk\right]$, i.e. $p = 1\,/\left(E\left[Bk\right] + 1\right)$ [22]. Assuming the p-persistent protocol yields to closely approximate the channel occupation resulting from the activity of R asymptotic mobile hosts, when $R \gg 1$ holds [22]. However,it does not provide accurate results with respect to the MAC delay experienced by a particular mobile host (i.e., the average time required by that mobile host to start a successful transmission). Based on these observations, in our hotspot model the tagged mobile host uses the standard IEEE 802.11 protocol, since its energy consumption significantly depends on the MAC delay it experiences (see Equation 7.13). However, to model the impact of background mobile

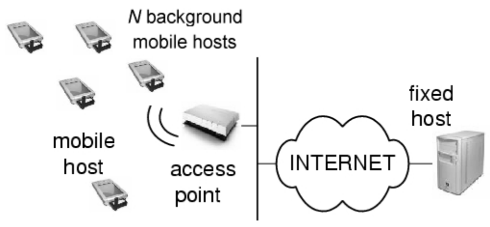


Figure E.1.: Wi-Fi hotspot scenario.

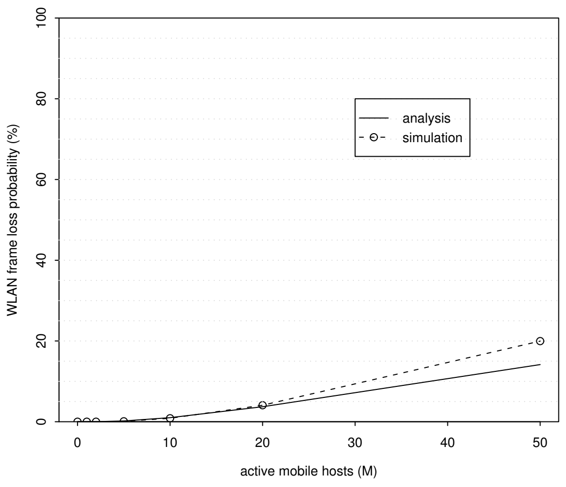


Figure E.2.: p_{loss} as a function of the number of active mobile hosts in the hotspot.

hosts on the energy consumption of the PSM mobile host, it is sufficient considering the channel occupation resulting from their activity. Therefore, we assume that the background mobile hosts use the p-persistent protocol, as this approximation greatly simplifies the analysis. The comparison between analytical and simulation results show that this approximation does not compromise the analysis reliability. To point out the dependence of p on the number of active mobile hosts ($M+1$ in our case), we hereafter refer to p as p_M.

Finally, we neglect frame disruptions due to transient channel fading and interference. Furthermore, we assume that frames sent by the PSM mobile host get never lost, i.e., they are successfully delivered within the maximum number of (re-)transmissions allowed by the MAC protocol. This assumption relies on the the retransmission policy of the 802.11 MAC protocol, that makes the data link service quasi-reliable. To corroborate this hypothesis, let us define p_{loss} as the probability that a PSM-mobile host frame is discarded after being (re-)transmitted for the maximum number of times. Figure E.2 plots p_{loss} as a function of the number of active mobile hosts in the hotspot, i.e., M. In this figure we show the results provided by both the analytical and the simulation models (a closed form of p_{loss} is provided below).

E.2. Modeling the MAC delay

Before deriving $E[tmac]$ we need to introduce the following definitions.

Equivalent-slot time (t_{sl}^{e}) . The time required by the tagged mobile host to decrement the backoff counter by one during the backoff procedure. When no other mobile hosts are active, t_{sl}^{e} is equal to the length of a slot (throughout referred to as t_{sl}). Otherwise it increases, due to transmissions of background mobile hosts that freeze the backoff procedure.

Collision time ($tcoll$) . The time during which the tagged mobile host cannot access the channel when a frame sent by the tagged-mobile host undergoes collision.

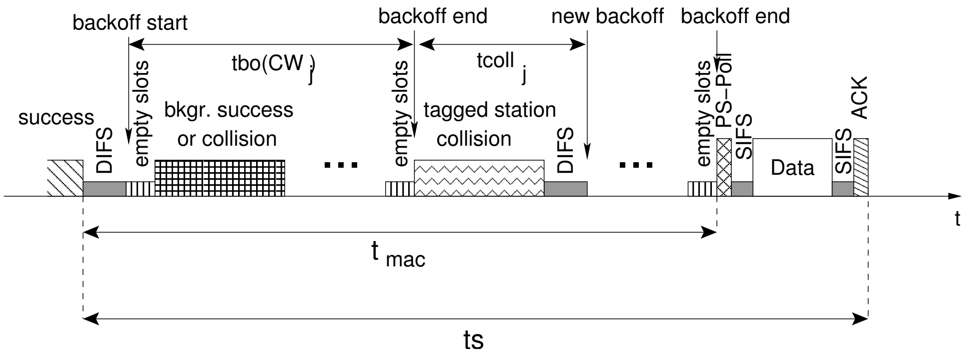


Figure E.3.: Snapshot of the time required to donwload a TCP segment ($ts_j^{(i)}$)

Backoff time ($tbo\,(CW)$) . The time required by the tagged mobile host to complete the backoff procedure when the contention window size is CW.

Retransmission limit (MAX) . The maximum number of (re-)transmissions allowed by the standard 802.11 MAC protocol before discarding a frame.

Free probability (p_f) . The probability that no other mobile hosts transmit in the same time slot used by the tagged mobile host to start a transmission, i.e., $p_f = (1 - p_M)^M$.

Loss probability (p_{loss}) . The probability that a frame is discarded after being (re-)transmitted MAX times without success, i.e., $p_{loss} = (1 - p_f)^{MAX}$.

Figure E.2 gives an example of the MAC protocol evolution during a generic TCP-segment donwload. This helps understanding analysis of $tmac$. As shown in Lemma 8, $tmac$ starts at the beginning of the first free slot after the successfull delivery of a frame. Therefore, at the beginning of $tmac$, a DIFS interval occurs during which all mobile hosts refrain from transmitting. Then, the backoff and the DCF procedures are executed in the standard way [35]. Since in our model we assume that the tagged frame is successfully sent within MAX attempts, we can derive the distribution of $tmac$ conditioned to i) experiencing i collisions before successfully delivering the frame; and ii) successfully delivering the frame within MAX attempts. Specifically, the following lemma holds.

Lemma 14 *The distribution of $tmac$ conditioned to experiencing i collisions, and to successfully delivering the frame within MAX, attempts is:*

$$tmac|_{i,MAX} = DIFS + \sum_{j=1}^{i+1} tbo\,(CW_j) + \sum_{j=1}^{i} tcoll_j \qquad \frac{(1-p_f)^i \cdot p_f}{1 - p_{loss}}, \quad i = 0, \ldots, MAX - 1 \;. \quad \text{(E.1)}$$

Proof. As noted before, in our model $tmac$ starts with a DIFS interval. Provided that the tagged mobile host experiences i collisions, it performs $i + 1$ backoff procedures. Based on [35], the the contention window is doubled after each collision, up to a maximum value. Then, it is kept constant at that maximum value. The contribution to $tmac$ of the first i attempts (i.e., when a collision occurs) is the time required to perform the backoff procedure ($tbo\,(CW)$) plus the time during which the collision is ongoing ($tcoll$)[1]. The contribution of the last attempt (i.e., when the frame is successfully delivered) is just the time required to perform the backoff procedure. Based on this observations, the expression of $tmac|_{i,MAX}$ follows immediately. As far as the probability distribution of $tmac|_{i,MAX}$, we can use the following line of reasoning. If p_{MAX} denotes the probability of delivering a frame successfully within MAX attempts, and p_i denotes the probability of having exactly i collisions, the conditional probability $p_i|_{MAX}$ is equal to p_i/p_{MAX}. The closed form provided in Equation E.1 can be derived by recalling that, since we have assumed a p-persistent MAC protocol: i) p_i is equal to $(1-p_f)^i \cdot p_f$; and ii) p_{MAX} is equal to $1-p_{loss}$. ■

As shown in Equation E.1, $tmac|_{i,MAX}$ depends on both $tbo\,(\textsc{cw})$, and on $tcoll$. In order to derive a closed form expression of $E\,[tmac]$, it is worth analyzing these components in isolation.

E.2.1. Analyzing the time spent due to collisions ($tcoll$)

By definition, $tcoll$ is the interval elapsed from the instant when the tagged mobile host ends a backoff procedure, up to the instant when it starts the next backoff procedure, when a collision occurs. Let us focus on a particular collision, and let us assume that C background mobile hosts collide with the tagged mobile host. Furthermore, let $frSz_i$, $i=1,\ldots,C$ be the size of the frame sent by the i-th mobile host during the collision. Then, the following lemma holds.

Lemma 15 *tcoll can be expressed as:*

$$tcoll = \tau + \frac{phyHdrSz}{phyR} + \frac{macHdrSz}{baseR} + \max_i \left\{ \frac{frSz_i}{dataR} \right\} + DIFS\ . \tag{E.2}$$

Proof. In our model the frames sent by the tagged mobile host that may undergo collisions are either PS-Poll frames or Data frames containing TCP acks. Furthermore, we assume that the payloads of Data frames sent by background mobile hosts are longer than TCP ACKs. Hence, when a collision occurs, the tagged mobile host transmits the frame and immediately senses the medium as busy, since the colliding mobile hosts are still transmitting (recall that the PS-Poll frame is just composed by the MAC header and the FCS field). The overall time during which the medium remains busy is equal to the time interval required to transmit the longest Data frame. When the medium becomes free it remains idle for a DIFS interval, due to the MAC protocol definition. Finally, the tagged mobile host starts the next backoff procedure. From this line of reasoning, deriving Equation E.2 is straightforward. ■

For the sake of simplicity, in our model we assume that the Data frames sent by background mobile hosts are of the same, constant size (throughout referred to as FS). Hence, $tcoll$ becomes a constant term, equal to

$$tcoll = \tau + \frac{phyHdrSz}{phyR} + \frac{macHdrSz}{baseR} + \frac{FS}{dataR} + DIFS\ . \tag{E.3}$$

[1]As shown in the following, $tcoll$ includes also DIFS intervals that occur after the collision is over.

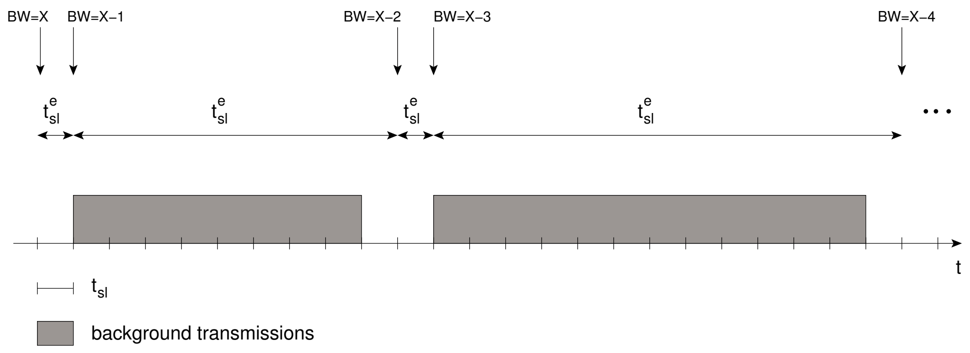


Figure E.4.: Backoff procedure: evidence on the equivalent-slot time.

E.2.2. Analyzing the time spent in backoff ($tbo\,(CW)$)

By definition, $tbo\,(CW)$ is the time required by the tagged mobile host to execute the backoff procedure when the contention window is CW. The first step of the backoff procedure is choosing a value between 0 and $CW-1$, according to a uniform distribution. This value (hereafter referred to as $X\,(CW)$) represents the number of *free* slots the tagged mobile host must wait before starting the next transmission attempt. It is worth noting that, during the backoff procedure, background mobile hosts may transmit frames, as usual. This causes the tagged mobile host to freeze the procedure until the channel returns free. To analyze this behavior, it is worth recalling the definition of *equivalent-slot time* (see Figure E.4). The equivalent-slot time is a random variable that measures the time required by the tagged mobile host to decrement by one its backoff counter. When a slot remains free (i.e., background mobile hosts do not transmit in that slot), the value of the equivalent-slot time equals the value of a slot (i.e., $t_{sl}^{e}=t_{sl}$). Otherwise, it increases, due to ongoing transmissions on the channel.

Based on the equivalent-slot time defintion, $tbo\,(CW)$ can be seen as the time required for $X\,(CW)$ equivalent-slot times to elapse, i.e., $tbo\,(CW)=\sum_{i=1}^{X(CW)} t_{sl\,i}^{e}$. Based on these observation, it is possible deriving the average value of $tbo\,(CW)$ as follows:

Lemma 16 *The average value of $tbo\,(CW)$ is*

$$E\left[tbo\,(CW)\right]=E\left[X\,(CW)\right]\cdot E\left[t_{sl}^{e}\right]\ . \tag{E.4}$$

Proof. Equation E.4 can be derived by recalling that: i) $X\,(CW)$ is sampled from a uniformly distributed random variable, which is independent of $\left\{t_{sl\,i}^{e}\right\}_{i}$; and ii) the random variables $\left\{t_{sl\,i}^{e}\right\}_{i}$ are independent on $X\,(CW)$ since the length of an equivalent-slot time depends only on the activity of the background mobile hosts. ■

The last step to provide a closed form of $E\left[tbo\,(CW)\right]$ is deriving closed form expressions of $E\left[X\,(CW)\right]$ and $E\left[t_{sl}^{e}\right]$. By recalling that $X\,(CW)$ is a random variable uniformly distributed between 0 and $CW-1$, $E\left[X\,(CW)\right]$ is as follows:

$$E\left[X\,(CW)\right]=\frac{CW-1}{2}\ . \tag{E.5}$$

On the other hand, to evaluate the average value of t_{sl}^{e}, we can use the following line of reasoning. Let us focus on the portion of an equivalent-slot time during which the channel is busy due to an ongoing transmission of background mobile host(s). During this time interval two events may occur, i.e., i) a successfull transmission; or ii) a collision among background mobile hosts. If tw denotes a random variable measuring the length in time of this time interval, it can be shown that tw is distributed as follows[2]:

$$\begin{cases} 2 \cdot \tau + 2 \cdot \frac{phyHdr}{phyR} + \frac{macHdr+ackSz}{baseR} + \frac{FS}{dataR} + SIFS + DIFS & (1-p_M)^{M-1} \\ \tau + \frac{phyHdr}{phyR} + \frac{macHdr}{baseR} + \frac{FS}{dataR} + DIFS & 1-(1-p_M)^{M-1} \end{cases} . \quad \text{(E.6)}$$

Hereafter, tw is also referred to as “waiting time". Based on the definition of tw, an equivalent slot can be made up of an arbitrary number of waiting times, followed by a free slot. Indeed, due to the assumption used to model the hotspot, after a waiting time is elapsed, there are still M mobile hosts trying to access the channel, and hence another waiting time may start. Based on these observations, we can derive the average value of t^e_{sl} as in the following lemma.

Lemma 17 *The length in time of an equivalent slot, conditioned to having k waiting time is*

$$t^e_{sl}|_k = \sum_{i=1}^{k} tw_i + t_{sl} \; . \quad \text{(E.7)}$$

Furthermore, the average value of t^e_{sl} is as follows:

$$E\left[t^e_{sl}\right] = \frac{1-p_f}{p_f} \cdot E\left[tw\right] + t_{sl} \; , \quad \text{(E.8)}$$

where $E\left[tw\right]$ is the average value of the waiting-time distribution.

Proof. Equation E.7 derives immediately by the above observations. Futhermore, the average value of $t^e_{sl}|_k$ is equal to $k \cdot E\left[tw\right] + t_{sl}$. The average value of the equivalent-slot time ($E\left[t^e_{sl}\right]$) can be derived as $\sum_{k=0}^{\infty} E\left[t^e_{sl}|_k\right] \cdot p(\text{k waiting times})$. Furthermore, it is easy to shown that $p(\text{k waiting times})$ follows a geometric law with parameter $(1-p_f)$, i.e., $p(\text{k waiting times}) = (1-p_f)^k \cdot p_f$. The closed form of $E\left[t^e_{sl}\right]$ can be derived after simple manipulations. ■

By substituting Equations E.5 and E.8 into Equation E.4, we can finally derive a closed form of $E\left[tbo\left(\textsc{cw}\right)\right]$.

E.2.3. Evaluating the average MAC delay ($E\left[tmac\right]$)

We are now in the position of deriving a closed form of $E\left[tmac\right]$. To be more precise, since we have assumed that the tagged mobile host successfully delivers a frame within MAX attempts, in the following we derive $E\left[tmac\right]$ conditioned to having a successfull transmission within MAX attempts, i.e., $E\left[tmac|_{MAX}\right]$. Specifically, the following theorem holds.

Theorem 8 *Let*

- ✧ *$S_N\left(q\right)$ and $Q_N\left(q\right)$ be the well-known closed-form expressions for the sequences $\sum_{i=0}^{N} q^i$ and $\sum_{i=0}^{N} i \cdot q^i$, respectively;*

[2]In detail, Equation E.6 holds if we assume that collided mobile hosts can start a new DCF and backoff procedure after sensing the channel idle for a DIFS interval instead of an EIFS interval.

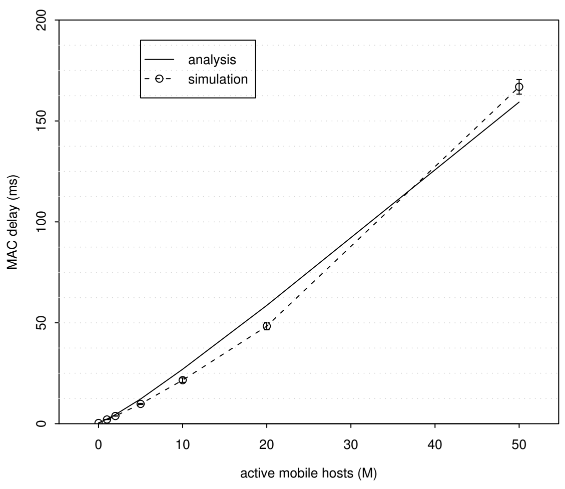


Figure E.5.: Validation of $E\left[tmac\right]$.

- ✧ *CW_{min} be the minimum contention-window size allowed by the standard IEEE 802.11 MAC protocol, measured in number of slots.*

Then the average value of the MAC delay, conditioned to successfully delivering the frame within MAX attempts, is

$$\begin{aligned}E\left[tmac|_{MAX}\right] &= DIFS + \frac{p_f}{1-p_{loss}} \cdot \{tcoll \cdot Q_{MAX-1}\left(1-p_f\right) + \\ &+ \frac{E\left[t_{sl}^e\right] \cdot CW_{min}}{2} \cdot \left(2 \cdot S_{MAX-1}\left(2 - 2 \cdot p_f\right) - S_{MAX-1}\left(1-p_f\right)\right) + \\ &- \frac{E\left[t_{sl}^e\right]}{2} \cdot \left(Q_{MAX-1}\left(1-p_f\right) - S_{MAX-1}\left(1-p_f\right)\right)\Big\} \ .\end{aligned} \tag{E.9}$$

Proof. $E\left[tmac|_{MAX}\right]$ can be evaluated as $\sum_{i=0}^{MAX-1} E\left[tmac|_{i,MAX}\right] \cdot p(\text{i collisions})$. Furthermore, from Equation E.1, $E\left[tmac|_{i,MAX}\right]$ can be written as

$$E\left[tmac|_{i,MAX}\right] = DIFS + \sum_{j=1}^{i+1} E\left[tbo\left(cw_j\right)\right] + i \cdot tcoll \tag{E.10}$$

Finally, Equation E.9 can be derived after simple manipulations by exploiting Equations E.10, E.4, and by recalling that Equation E.1 also provides a closed form of $p(\text{i collisions})$. ■

As a final remark, Figure E.5 plots the average MAC delay predicted by Equation E.9, and experienced by our simulator. Figure E.5 assesses the accuracy of the MAC-delay analytical model.

F. Impact of Burst Size on 802.11 Energy Consumption

This Appendix contains the detailed derivation of Equation 7.18, i.e.

$$E_{PSM} = a \cdot \mu \cdot K_{PSM} \ ,$$

where a is the scaling factor applied to the samples of the Web-page size distribution, and K_{PSM} is independent of a.

From Equation 7.16, E_{PSM} is as follows:

$$E_{PSM} = E\left[T\right] \cdot P_{sl} + E\left[T_{ac}\right] \cdot \left(P_{ac} - P_{sl}\right) \ .$$

From Equation 7.7, $E\left[T\right]$ is[1]:

$$E\left[T\right] = \frac{E\left[N_{BR}\right] \cdot E\left[BR\right]}{\gamma_{TCP}} = a \cdot \mu \cdot \frac{E\left[N_{BR}\right]}{\gamma_{TCP}} \triangleq a \cdot \mu \cdot K_1 \ ,$$

where K_1 is independent of $a \cdot \mu$.

Furthermore, from Equation 7.15, $E\left[T_{ac}\right]$ is as follows:

$$E\left[T_{ac}\right] = \frac{E\left[N_{BR}\right] \cdot a \cdot \mu}{MSS} \cdot \left(E\left[ts\right] + E\left[ta\right]\right) + \frac{E\left[T\right]}{BI} \cdot tb \triangleq a \cdot \mu \cdot K_2 + \frac{E\left[T\right]}{BI} \cdot tb \ ,$$

where K_2 is independent of $a \cdot \mu$.

As $E\left[T\right]$ is proportional to $a \cdot \mu$, $E\left[T_{ac}\right]$ can be expressed as:

$$E\left[T_{ac}\right] = a \cdot \mu \cdot K_2 + \frac{a \cdot \mu \cdot K_1}{BI} \cdot tb \triangleq a \cdot \mu \cdot K_3 \ ,$$

where K_3 is independent of $a \cdot \mu$.

Finally, E_{PSM} can be expressed as:

$$E_{PSM} = a \cdot \mu \cdot K_1 \cdot P_{sl} + a \cdot \mu \cdot K_3 \cdot \left(P_{ac} - P_{sl}\right) \triangleq a \cdot \mu \cdot K_{PSM} \ ,$$

where K_{PSM} is independent of $a \cdot \mu$. This concludes the proof.

[1] Recall that in this part of the analysis we have to set $E\left[UTT\right]$ to 0 in Equation 7.7 (see Section 7.5.2.1).

G. Impact of TCP throughput on 802.11 Energy Consumption

In this Appendix we prove that both E_{PSM} and E_{PSM} increase when either the TCP-segment loss probability (p_l^{tcp}) or the Round Trip Time (RTT) increase.

Let us focus on E_{PSM}. As in the previous Appendix, the most convenient expression for E_{PSM} in order to prove this claim is provided by Equation 7.16. Specifically, Equation 7.16 states that E_{PSM} is an increasing function of both $E\left[T\right]$ and $E\left[T_{ac}\right]$. In the following, we show that both these terms are increasing functions of either p_l^{tcp} and RTT.

The closed form expression of $E\left[T\right]$ is provided by Equation 7.7, which is reported here for the reader convenience:

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

The only term that is affected by p_l^{tcp} or RTT is γ_{TCP}. Specifically, from Equation 7.19, γ_{TCP} is a decreasing function of both p_l^{tcp} and RTT. Hence, when either of these parameters increase, $E\left[T\right]$ increases too. The rationale of this result lies in the very definition of T. Specifically, T is the time required to download N_{BR} bursts from the fixed server. When the throughput drops, the burst-download phases become longer, and T increases.

As far as $E\left[T_{ac}\right]$, by focusing on Equation 7.15 it can be noted that the only terms affected by γ_{TCP} is the component related to beaconing, i.e. $E\left[T\right]/\,BI \cdot tb$. Hence, also $E\left[T_{ac}\right]$ increases when γ_{TCP} drops.

Finally, it can be noted that E_{NO_PSM} is an increasing function of both p_l^{tcp} and RTT, as it is proportional to $E\left[T\right]$ (see Section 7.16). This concludes the proof.