Experimental Analysis of an Application-independent Energy Management Policy for Wi-Fi Hotspots

G. Anastasi•, M. Conti*, E. Gregori*, A. Passarella•

•Univ. of Pisa, Dept. of Information Engineering
Via Diotisalvi 2 - 56122 Pisa, Italy
{g.anastasi, a.passarella}@iet.unipi.it,

*CNR-IIT Institute
Via G. Moruzzi, 1 - 56124 PISA, Italy
{marco.conti, enrico.gregori}@iit.cnr.it

Abstract. In the near future more and more users will access Internet services by means of portable devices through wireless links. However, mobile computing is still strongly limited by the scarcity of energetic resources of portable devices. In this paper we propose and evaluate an application-independent energy management policy for a Wi-Fi hotspot scenario. Unlike the IEEE 802.11 Power Saving Mode, the proposed solution is able to adapt to the application traffic profile, thus saving a considerable amount of energy. For the same reason it is flexible, i.e., it exhibits good performance irrespectively of the specific network application, and even in the presence of concurrent applications. Experimental measurements performed on a prototype implementation with different traffic types have shown that our energy management policy is able to save up to 80% of the energy consumed in a legacy architecture, without a significant degradation on the QoS perceived by the user.

1. Introduction

In the next future more and more users throughout the world will access Internet services by means of mobile computers (notebooks, PDAs, smartphones, etc.) through wireless links. However, the potentialities of mobile computing are still strongly limited by the scarcity of energetic resources at mobile computers. Based on the past experience, and on projections on progresses in battery technology, in the near future it is wise to expect only small improvements in the battery capacity [7]. Therefore, it is important to manage energy efficiently.

Strategies for energy management have been investigated at several layers of the system architecture, including the hardware, the operating system [13,17], the network protocols [3,15,16], and the applications [12,14,18]. Experimental measurements have shown that the wireless network interface is responsible for a considerable fraction of the total energy consumption: up to 50% in small-size handheld computers [15]. Therefore, it is vital to design energy-efficient network protocols and applications. Energy management policies can be implemented at different layers of the network architecture. Wi-Fi cards include a Power Saving Mode (PSM) to reduce power consumption during idle periods. In a legacy TCP/IP architecture, Wi-Fi PSM allows to save up to 90% of the energy consumed by the wireless interface [16]. However, the PSM performance decreases significantly as the number of mobile computers served by the same Access Point increases [7]. In addition, PSM implements a *static* energy management policy, i.e., it is not able to adapt to the application behavior. In [16] an extension of PSM is proposed that is *adaptive* to the (Web) application traffic profile. An adaptive energy management policy is also proposed in [1]. [15] implements energy management at the transport layer by exploiting the Indirect-TCP model [9]. The wireless interface at the mobile computer is switched off whenever an *inactivity* timer expires, and resumed after a *sleeping* timeout. Both the inactivity and sleeping timeouts have *fixed values*. Since the application traffic profile varies dynamically, using fixed timeouts may result in poor performance. [3] too relies on an (extended) Indirect-TCP architecture, but suggests an *application-aware* approach to energy management. Application-aware energy management policies can be classified as *application-dependent* [2,4] and *application-independent* [5] policies. In the former case the energy management policy relies on some *a priori* knowledge of the application; in the latter case no *a priori* assumption about the specific application behavior is done.

In this paper we consider an application-independent policy. It monitors the traffic generated by the (non real-time) application(s) -- detecting idle phases and predicting their duration -- and manages the wireless interface of the mobile computer accordingly. Specifically, the mobile computer switches off its wireless interface when there are no more data to be exchanged over the wireless link. Since this energy management policy requires no a priori knowledge about the application behavior, it is suitable for any (non real-time) application. Furthermore, the proposed policy provides a standard socket interface: it can thus be used by legacy applications. Finally, it is

Work carried out under the financial support of the Italian Ministry for Education and Scientific Research (MIUR) in the framework of the Projects "Internet: Efficiency, Integration and Security" and FIRB-VICOM.

completely independent from the underlying wireless network technology.

In [5] a preliminary experimental analysis of the solution considered here, based on a single type of traffic (i.e., Web traffic), was provided. In this paper 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.

The paper is organized as follows. Section 2 describes the application-independent energy management policy. Section 3 reports the results of the experimental analysis. Finally, Section 4 concludes the paper.

2. Energy saving architecture

Figure 1 shows the reference network scenario considered in this paper. We refer to a typical Wi-Fi hotspot where the mobile computer connects to the fixed Internet infrastructure through the Access Point. Using the legacy TCP/IP architecture in such a scenario may result in a very bad energy utilization at the mobile computer [5]. Therefore, we exploited the *Indirect-TCP model* [9] and extended it by introducing optimizations and explicit energy management mechanisms. The resulting architecture is depicted in Figure 2.

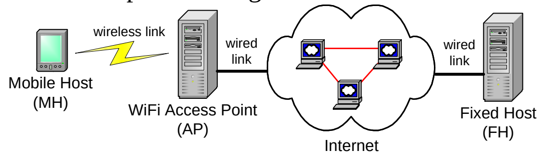


Figure 1. A WiFi-based Mobile Internet scenario

In the *Indirect-TCP model* the TCP connection between the Mobile Host (MH) and the Fixed Host (FH) is split at the Access Point (AP) into two TCP connections: one between the MH and the AP, and the other between the AP and the FH. A *Daemon* process at the AP relays data from one connection to the other. In our architecture a *Simplified Transport Protocol* (STP) is used in the wireless link instead of the TCP protocol. Unlike TCP, STP is tailored to the one-hop wireless environment and, hence, it provides a higher throughput [8], thus reducing the energy consumption.

Energy management mechanisms introduced in the extended *Indirect-TCP* architecture are based on the evidence that the MH does not exchange data continuously. On the other hand, data transfers are typically characterized by *bursts* separated by *idle* phases. Packets within the same burst are spaced by inter-arrival times. Our approach is based on the dynamic estimate of idle and inter-arrival times. By exploiting these estimates, it is decided when to switch off the wireless interface, and when to resume it. Hence, this policy works well if estimates are accurate, and approximates the “ideal energy management”, i.e., the ideal policy that knows in advance idle and interarrival times. The core of our energy management policy consists of a set of algorithms for deriving the above estimates. We integrated these algorithms in the extended *Indirect-TCP* architecture by introducing the *Power Saving Packet Transfer* (PS-PT) protocol on top of the STP protocol (Figure 2). Finally, the *Socket Handler (SH)* layer provides a standard socket interface to applications.

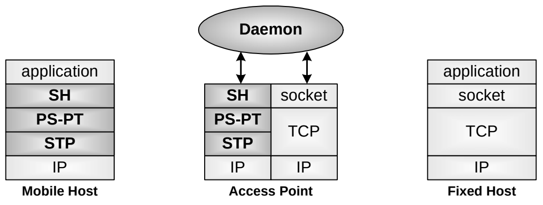


Figure 2. Energy Saving Architecture.

2.1. PS-PT Protocol Description

In this section we provide a brief description of the PS-PT protocol (a detailed description can be found in [5]). This protocol includes algorithms for estimating packet inter-arrival times and idle periods. We decided to implement these algorithms at the AP in order to relieve the MH from the related computational burden.

The PS-PT protocol is conceived as a simple master/slave protocol. When the Daemon at the AP (see Figure 2) receives a new packet destined for the MH, the PS-PT at the AP records the time instant at which the packet arrived. Similarly, at the MH, the PS-PT adds a timestamp to each packet sent to the AP. This timestamp records the time instant at which the packet arrived at the PS-PT module. The PS-PT at the AP uses the above information for estimating packet inter-arrival times and idle period lengths. In particular, when the transmission queues at both sides are empty, the PS-PT at the AP estimates the time instant when the next packet is expected to arrive. Based on this estimate it decides whether to switch off the wireless interface at the MH, or not. It is worthwhile to recall that the wireless interface has a transient in getting on, t_{PWON}, during which it drains energy from the battery but is not available for exchanging data. Therefore, for interarrival and idle times smaller or comparable with t_{PWON} it is more convenient to leave the network interface on. If the PS-PT protocol decides to switch off the wireless interface, it sends a *shutdown* command to the MH including the duration of the disconnection period. The MH switches off the wireless interface and sets up a timer (*disconnection timer*) by using the notified disconnection period. While the MH is disconnected, packets destined for it are stored at the PS-PT of the AP. Upon expiration of the disconnection timer, the MH polls the AP and receives either new packets or an updated estimate for the disconnection period.

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).

We defined a Power Saving Index and a QoS index. The Power Saving Index, *I_ps*, is as follows

$$I\_ps = \frac{\text{consumption in our architecture}}{\text{consumption in the legacy architecture}}\,. \qquad \textbf{(1)}$$

I_ps measures the percentage of energy consumed by our architecture with respect to the legacy one, and, hence, provides an indication of the energy saving achieved by using our solution. Since in a Wi-Fi hotspot the power drained by the wireless interface of a MH is almost independent on its status (receiving, transmitting or idle), the energy consumption is roughly proportional to the overall time the wireless interface remains on. Therefore, to derive the energy consumption we measured this time interval, both in our architecture and in the legacy architecture.

To evaluate the impact on the QoS we measured the additional delay introduced by the energy management policy in our architecture (with respect to the legacy architecture) to complete a network activity (e.g., the download of a Web page or a mail check).

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.

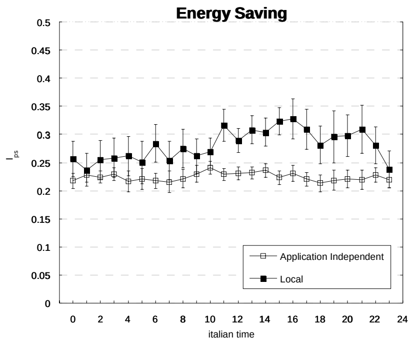
3.1. Scenario 1: Web Traffic

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 [10] 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.

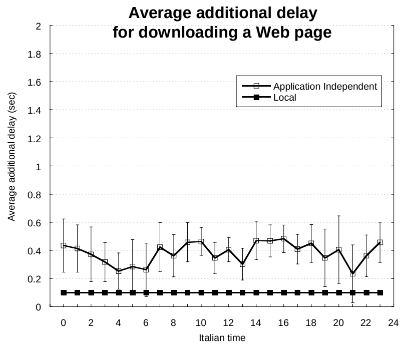
We performed a large number of experiments. Each experiment included 100 page-transfer operations from the Web server to the client (an experiment stopped when the whole page "in flight" arrived at the client). In each experiment, the same set of pages were requested in parallel both in our architecture, 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 3 (a) shows the *I_ps* index as a function of time. It clearly emerges that our architecture allows 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 a very simple energy management policy, hereafter referred to as *local*. 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 called "local" because it can be implemented by only exploiting information available *locally* at the Web browser. Unlike the application-independent 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 3 (b) shows the average additional delay introduced in downloading a Web page with respect to the legacy architecture. The additional delay introduced by the local policy is constant and almost negligible (it is due to the time needed to switch on the wireless interface upon receiving a new user request). However, also the additional delay introduced by the application-independent policy 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 (see [5]). Based on the above results we can conclude that the QoS degradation introduced by the application-independent policy can be considered as acceptable for Web-browsing applications.



(a)



(b)

Figure 3. *I_ps* as a function of time (a). Average additional delay as a function of time (b).

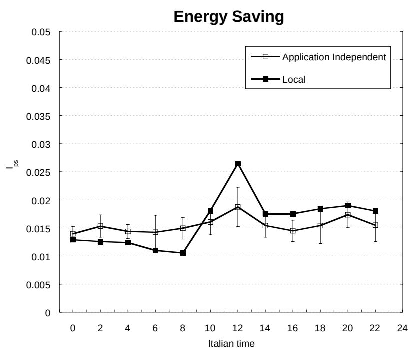
3.2. Scenario 2: E-mail Traffic

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.

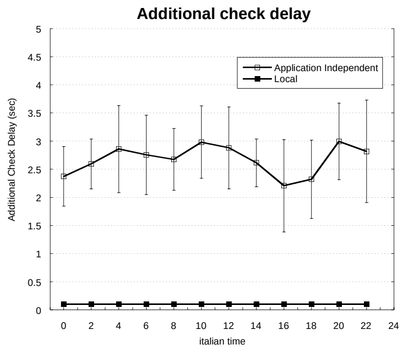
E-mail involves two protocols: POP3 for downloading messages from the POP server to the user's computer (the MH in our case), and SMTP for sending messages from the user's computer to the SMTP server that, in its turn, will forward the same messages to the final destination. To perform our experiments 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 [11]. 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 4 (a) shows the I_{PS} index as a function of time for the application-dependent and local policies, respectively. Figure 4 (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 data-transfer time (typically in the order of seconds). Therefore, reducing energy consumption even in the data-transfer phase does not produce a significant effect.

The application-independent policy introduces an average additional delay that is typically in the order of 2-3 sec. (the 90th percentile is less than 10 sec), while the additional delay introduced by the local policy is, as above, constant and almost negligible. 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 shown in [6], this has an impact on the additional delay introduced. Though the additional delay for the application-independent policy is significantly greater than that related to the local policy, 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.



(a)



(b)

Figure 4. I_{ps} as a function of time (a). Average additional delay as a function of time (b).

3.3. Scenario 3: 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).

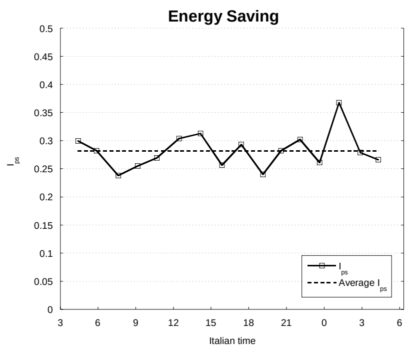
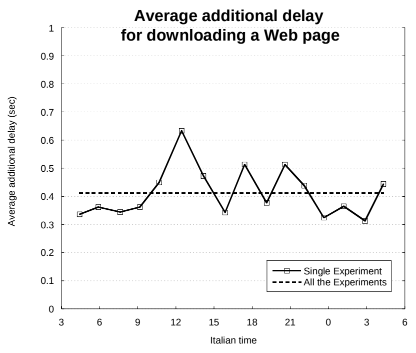
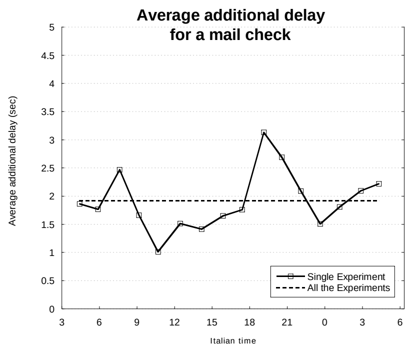


Figure 5. I_{PS} as a function of time.



(a)



(b)

Figure 6. Average additional delay for downloading a Web page (a) and for a mail check (b).

Figure 5 shows the I_{PS} index as a function of time in this scenario. Even in a mixed-traffic 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 3 (a)). 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 mixed-traffic 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 e-mail traffic. This results in a lower additional delay.

4. Conclusions

In this paper we have evaluated through an experimental analysis an energy management policy conceived for a Wi-Fi hotspot scenario. The proposed solution relies on the Indirect-TCP model and is application independent. We have implemented our energy management policy in a prototype system, and extensively evaluated it in the presence of different traffic types. The experimental results have shown that our application-independent policy is both flexible and efficient. It is able to save a considerable amount of energy (with respect to a legacy architecture without energy management), irrespectively of the specific applications. Even in the mixed-traffic (Web + e-mail) scenario the energy saving achieved is, on average, of 72%. In addition, this energy saving is obtained without a significant impact on the QoS perceived by the user.

References

[1] M. Anand, E. Nightingale, J. Flinn, “Self-Tuning Wireless Network Power Management”, Proc. *ACM Mobicom 2003*, S. Diego (CA), September 2003.

[2] G. Anastasi, M. Conti, E. Gregori and A. Passarella, “A Power Saving Architecture for Web Access from Mobile Computers”, Proc. *IFIP Networking 2002*, Pisa (I), May 2002, LNCS 2345, pp. 240-251.

[3] G. Anastasi, M. Conti, W. Lapenna, “A Power Saving Network Architecture for Accessing the Internet from Mobile Computers: Design, Implementation and Measurements”, *The Computer Journal*, Vol. 46, N. 1, January 2003, pp. 3-15.

[4] 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.

[5] G. Anastasi, M. Conti, E. Gregori, A. Passarella, “Balancing Energy Saving and QoS in the Mobile Internet: an Application-Independent Approach”, Proc. *Hawaii International Conference on System Science (HICSS-36)*, Hawaii, January 2003.

[6] G. Anastasi, M. Conti, E. Gregori, A. Passarella, “A Performance Study of Power-Saving Polices for WiFi Hotspots”, *Computer Networks*, Vol. 45, No. 2, June 2004.

[7] G. Anastasi, M. Conti, E. Gregori, A. Passarella, “Saving Energy in WiFi Hotspots through 802.11 PSM: an Analytical Model”, Proc. *Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt 04)*, Cambridge (UK), March 2004.

[8] H. Balakrishnan, V. Padmanabhan, S. Seshan, R. Katz, “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links”, *IEEE/ACM Trans. on Networking*, Vol. 5, N.6, December 1997.

[9] A. Bakre, B. R. Badrinath, “Implementation and Performance Evaluation of Indirect TCP”, *IEEE Trans. on Computers*, Vol.46, No.3, March 1997.

[10] P. Barford e M. Crovella, “Generating Representative Web Workloads for Network and Server Performance Evaluation”, *Proc. of ACM SIGMETRICS ´98*, Madison (WI) pp. 151-160, June 1998.

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

[12] J. Flinn, S. Y. Park and M. Satyanarayanan, “Balancing Performance, Energy, and Quality in Pervasive Computing”, Proc. of *IEEE Int. Conf. on Distributed Computing Systems* (*ICDCS 2002*), Wien (A), July 2002.

[13] D.P. Helmbold, D.E. Long, B. Sherrod, "A Dynamic Disk Spin-down Technique for Mobile Computing", Proc. *ACM Mobicom’96*, New York, Nov. 1996, pp. 130 – 142.

[14] A. Joshi, “On Proxy Agents, Mobility, and Web Access”, *ACM/Baltzer Mobile Networks and Applications*, Vol. 5 (2000), pp. 233-241.

[15] R. Kravets, P. Krishnan, “Power Management Techniques for Mobile Communication”, Proc. *ACM Mobicom’9*8, Dallas (Texas), October 1998.

[16] R. Krashinsky , H. Balakrishnan, “Minimizing Energy for Wireless Web Access with Bounded Slowdown”, Proc. *ACM Mobicom 2002*, Atlanta (Georgia), September 2002.

[17] J. R. Lorch, A. J. Smith, “Software Strategies for Portable Computer Energy Management”, *IEEE Personal Communications*, Vol. 5, N.3, June 1998.

[18] E. Pitoura, G. Samaras, “Data Management for Mobile Computing ”, Kluwer Academic Publishers, 1998.