A Survey of Software Based Energy Saving Methodologies for Handheld Wireless Communication Devices

Kshirasagar Naik

Tech. Report No. 2010-13
Dept. of ECE, University of Waterloo

***Abstract*— Advances in computing hardware, communications technologies, and novel multimedia applications are spurring the development of smart phones and personal digital assistants. There is world-wide accessibility to 2G, 3G, and Wi-Fi networks. Some examples of well-known wireless handheld devices are BlackBerry, iPhone, iPad, iPod, and Kindle. On the one hand, small size and light weight are their attractive features for high mobility and accessibility. On the other hand, the same features impose significant constraints on their processing, memory, and energy storage capabilities, thereby limiting the device's general functionalities and availabilty. User expectations in terms of performance from handheld devices are ever increasing. In addition to performance expectations, the requirement of *portability* imposes severe constraints on *size* and *weight* of a handheld system. Consequently, batteries too are small and light, and, therefore, the system energy budget is severely limited. The amount of energy in a fully charged battery is one of the important resources of a handheld system, and battery lifetime is an important characteristic. Unfortunately, improvements in energy density of batteries have not kept pace with the advancements in microelectronics technology.**

Therefore, researchers have proposed novel architectures, strategies, methodologies, and techniques to make handheld devices energy efficient. In this paper, we study the results published in about 200 reserach papers to have a comprehensive understanding of the various approaches to make handheld devices energy efficient. We present our study in the form of 15 cohesive sections that include the following topics: related surveys, smart batteries, energy-efficient graphical user interface design, the concept of a sleep state for an operating system, power efficient communication, proxy assisted energy saving, source-level power control, transport control protocol based energy saving, upper-level power management, virtual memory on the network, programming and compilation techniques, integrated power management, energy estimation models, impacts of dynamic power management (DPM) strategies, and future directions in energy saving.

I. Introduction

The past decade has witnessed tremendous growth in the popularity of the Internet and wireless handheld devices. For wireless Internet access, there is almost universal coverage with 2G, 3G, and Wi-Fi networks. The commonly used handheld devices are smart phones and PDAs (personal digital assistants), with Internet browing capability. Some of the popular ones are *iPAQ*, *BlackBerry*, *iPhone*, *iPod*, *iPad*, and *Kindle*. Application specific handheld devices, such as *iPod* for music and *Kindle* for electronic book reading, are gaining much popularity. The recently released *iPad* falls in between a smart phone and a laptop, in terms of functionalities, processing capabilities, and mobility, while supporting thousands of third-party developed applications.

K. Naik is with the Dept. of ECE, University of Waterloo, Waterloo, Ontario, Canada, N2L3G1 (Email: snaik@uwaterloo.ca).

Today's handheld devices come with a number of wireless interfaces, namely, 2G, 3G, Wi-Fi, and Bluetooth, for a variety of applications. The 3G interface is used outdoor, whereas the Wi-Fi interface is for indoor applications. The Bluetooth interface is for shortrange communications between a handheld device and another personal device. Wi-Fi networks support high data rates in small regions, with limited mobility, whereas 3G networks support lower data rates in wide regions and with high mobility.

Handheld devices have been offering faster processors, more memory, multiple radio interfaces, and powerful operating systems (OS), and, consequently, heavier applications. Moreover, specialized operating systems have been developed for handheld devices. Microsoft's *Windows Mobile*, Google's *Android*, and *LiMo* (Linux Mobile) are example operating systems designed for mobile devices. Some popular applications on mobile devices are voice calls, email access, Internet browsing, video games, and video playing.

With advances in microelectronics, there is every effort, subject to size constraint, to make a handheld device appear like a laptop computer in terms of application delivery. In addition to performance expectations, the requirement of *portability* imposes severe constraints on the *size* and *weight* of a handheld system. Portable devices commonly run on rechargeable batteries to support user mobility. The *small size* and *light weight* requirements of a handheld device imply that its battery be proportionately small in volume. Consequently, the system energy budget is severely limited [5]. A battery must be charged before its remaining energy falls below a threshold level to keep the device running. Battery charging limits the mobility of users and the usability of the device. The full charge of a battery is one of the key resources, and battery lifetime is an important characteristic of handheld devices [39]. While computing and communication capabilities of handheld devices have increased by orders-of-magnitude in the past two decades, battery energy density has only *tripled* in the same period of time [2].

Therefore, hardware and software designers have adopted a variety of methodologies and techniques to reduce the amount of energy drawn from the battery. These methodologies and techniques include low-power electronics, smart

batteries [3] to report their state of charge (SoC) [4] to operating systems and applications, power-saving modes of operation of processors, wireless network interface cards, and displays, efficient algorithm design and implementations, and efficient graphical user interface (GUI) design. The concepts of proxies between servers and handheld devices and source level (i.e. server level) media stream control have been introduced to make handheld devices energy efficient. Researchers have worked on TCP-based energy saving in handheld devices. A number of techniques have been introduced at the application level to save energy. In order to run applications on handheld devices with virtual memory requirements, energy efficient network-based swapping has been proposed. At the coding and compilation level, techniques have been developed to reduce the energy cost of programs. Finally, system models have been developed to predict the energy cost of applications. In this paper, we review the main research results published along those directions.

We remind the reader that our focus is on handheld wireless devices and not individual sensors. We do recognize that much work have been done in routing message in sensor networks to make sensors energy efficient [64]. However, we will exclude those results from this paper. Similarly, we will exclude the techniques for energy saving in wall-powered personal computers [75]. We have organized this paper on a section-by-section basis as follows:

Section II. Related Surveys: In this section, we will discuss the scopes of four survey papers that have appeared in the literature over the past ten years, without going into their details, in order to avoid overlaps. Readers with specific interest in the subject matters of those surveys may refer to those papers for additional details.

Section III. Smart Batteries: Smart batteries play a key role in making applications adaptive to the amount of energy left in the battery, that is its SoC, and its present rate of energy draining. We discuss the idea of smart batteries in Section III.

Section IV. Energy-Efficient GUI Design: In this section we summarize the strategies and techniques that have been developed to make displays, user interactions, and graphical user interfaces (GUI) energy efficient. We will provide an understanding of human-computer interactions and a process for predicting user interaction time and energy.

Section V. Sleep to Save Energy: The architectural details of adding a *sleep* state to a processor and the concept of *wake-on-wireless* have been discussed in Section V. The former concept is useful in saving energy while applications are idle, while the latter saves energy in the communication module of a hand-held device.

Section VI. Power Efficient Communication: In this section we summarize the power saving mode of the IEEE 802.11 series of medium access control (MAC) protocols, newly proposed power saving techniques for IEEE 802.11 based hand-held devices, MAC-level download scheduling by Access Points (AP), and power management in WiMAX subscriber stations.

Section VII. Proxy Assisted Energy Saving: A proxy machine is a computing system located between user devices and their servers on the Internet. A proxy can enable a handhald device to save energy in a number of ways, for example: reduce the volume of contents to be downloaded by user devices and make downloaded data traffic bursty so that the intervals between bursts are long enough for devices to put their communication interfaces to sleep state. The capabilities of a *Transforming Proxy*, an HTTP-level *Power Aware Web Proxy*, a *Power Aware Streaming Proxy*, and a *Streaming Audio Proxy* to enable hand-held devices and laptops to save energy have been explained in this section.

Section VIII. Source-level Power Control: In this section we summarize the energy saving techniques applied at the source level, that is by the media servers. Examples of server-based techniques are: (i) traffic shaping to enable a user device to put its communication interface to sleep state; and (ii) resolution control of video frames. These techniques do not involve any proxy between the media servers and handheld devices.

Section IX. TCP Based Energy Saving: The Transmission Control Protocol (TCP) is widely used in web-based applications to transport data between clients and servers in a reliable manner. TCP-related energy efficieny of handheld devices are categorized into two groups: computational energy cost of TCP and TCP-assisted controlling the wireless network interface card (WNIC) for energy saving. In Section IX, we review the computational energy cost of TCP, TCP-based state control of WNIC, and the concept of a power saving transport protocol.

Section X. Upper-level Power Management: A number of techniques have been studied at the application-level to achieve energy efficiency in hand-held devices. Those are data compression and download scheduling at the application-level and computation offloading. In the first approach, the decompression tasks on a client device are appropriately interleaved with downloading activies to maximize energy saving. In the computation offloading approach, some computation-intensive tasks are migrated from a user device to a server. Details of those approaches and their energy saving potentials are discussed in Section X.

Section XI. Virtual Memory on the Network: In hand-held devices, storage memory is small or absent due to weight, size, and power constraints. In Section XI, we discuss the efficacy of implementing a network-based virtual memory system for handheld devices.

Section XII. Programming and Compilation Techniques: In this section we discuss a number of coding and compilation techniques to reduce the energy cost of executing a program. In addition, we provide the overview of an architecture for energy-aware programming.

Section XIII. Integrated Power Management: Power management has been studied both at component level and high level. In additional, several solutions for energy effi-

ciency have been proposed at various *computational* levels, namely, cache and external memory access optimization, dynamic voltage scaling, dynamic power management for disk and network interfaces, efficient compilers, and application/middleware adaptations. Some integrated approaches combining the energy saving potentials of multiple techniques have been summarized in Section XIII.

Section XIV. Energy Estimation Models: Energy constrained devices should be able to estimate the energy cost of an application to be executed. The capability to estimate the energy cost of an application can be utilized to make subsequent decision about its activities based on user input and sustainable battery life. Therefore, there is a need for robust energy cost models. In Section XIV, we discuss *low-level energy modeling*, *high-level energy modeling*, *system-level energy modeling*, and *characterization of applications*.

Section XV. Impact of DPM Strategies: In general, a DPM component is expected to raise the energy-related quality attributes of the system without impacting the systems original functionality. In other words, there is a need to verify that the added component does not interfere with the original functionality of the system, while, at the same time, making the system energy efficient. In this section, we explain how high-level system description languages, namely, *Aemilia*, *Real-time Maude*, and *Generalized Stochastic Petri Nets*, can be used to model and analyze systems with a DPM component.

Section XVI. Summary: This section provides a summary of the paper.

Section XVII. Future Directions: In this section, we discuss new opportunities to develop novel approaches to making handheld devices more energy efficient.

II. Related Surveys

Considering the profound need for energy saving and the tremendous progress made on the topic, survey papers have appeared in literature from time to time. In this section, we summarize two such papers: one about system-level DPM [196] and one about power-aware mobile multimedia applications [191]. Both the papers are very informative, and it is useful to get broad perspectives of those articles. We remind the reader that we will exclude energy saving in wireless sensor networks [64] and wall-powered personal computers [75].

A. Survey of Design Techniques for System-Level DPM [196]

Benini et al. [196] have surveyed system-level DPM approaches for the hardware portion designed with power manageable components (PMC). The key characteristic of a PMC is the availability of multiple *modes of operation* between the two extremes of high-performance high-power and low-power low-performance. Note that transitions between modes of operations have two associated costs, namely, *delay* (we denote the time required to enter and exit the low-power state by T_{TR}) and *power* (we denote the transition power cost by P_{TR}). It is the transition costs which may prevent system designers from frequently moving a system into its low power states. The concept of a *break-even time* (T_{BE})has been introduced to study the effectiveness of DPM policies because of the non-negligible cost of state transitions. The *break-even time* is defined as the minimum inactivity time required to compensate the cost of entering a power saving state. T_{BE} can be expressed as the sum of two components: (i) the time required to enter and exit the low-power state (i.e. T_{TR}); and (ii) the minimum time that must be spent in the low-power state to compensate the additional transition power P_{TR}.

General energy saving approaches are built upon the following premises:

- Systems experience *nonuniform* workloads while running an application. In other words, components remain idle from time to time: $idle(t_1) - busy(t_2) - \ldots idle(t_i) - busy(t_{i+1}) - \ldots$, where $idle(t_i)$ means a component remains idle for a duration of t_i.
- By observing the workload it is possible to predict its fluctuations with some confidence.

The structure of a generic power management mechanism has been illustrated in Fig. 1, where the *Observer* block collects information about the workload of all PMCs in the system, and the *Controller* implements DPM policies and issues commands to cause state transitions. For example, *timeout* is a simple policy that smartphones implement to turnoff their displays after a fixed inactivity period.

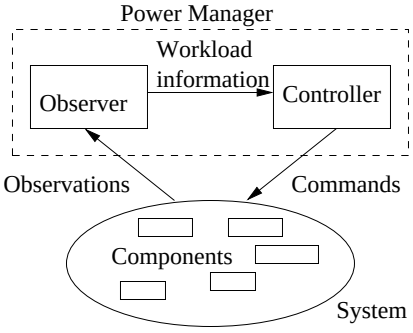


Fig. 1. Abstract structure of a system-level power manager [196].

Power management approaches have been classified into two broad classes as follows:

- *Predictive schemes:* Predictive schemes assume that components are essentially modeled as *two-state* systems: one active state (say, *On*) and one low-power state (say, *Off*). For the SA-1100 processor, the *On* state is a high-level state representing two low-level states, namely, *Run* and *Idle*, whereas the *Off* state corresponds to the *Sleep* state. Power management approaches falling in this category make the prediction, by observing the past history of the workload of a component, that the idle time of the component is highly likely to be greater than the break-even time in the near future: $T_{idle} > T_{BE}$. The predictive schemes are further divided into *static* and *adaptive* techniques. The effectiveness of predictive schemes in saving eneregy depends upon the

accuracy of workload prediction – and accuracy can be increased by designing specialized, workload-specfic predictors. Predictive schemes are heuristic and their parameter tuning is difficult to achieve, and those cannot finely control performance penalty.

- *Stochastic optimum control schemes:* These schemes can: (i) take advantage of the multiple low-power states of some components; (ii) better handle uncertainties in workload, state transition times, and modeling policy optimization. Consequently, policy optimization involves both *when* to make state transitions and *which* transition. Power management problems are studied with *controlled Markov processes* by assuming that workload can be modeled as Markov chains. Therefore, one can: (i) model the uncertainties in system power consumption and transition response times; (ii) model more complex systems with multiple power states, buffers, and queues; (iii) compute globally optimum power policies; (iv) and tradeoff between performance and power.

Finally, the survey discusses various DPM implementation techniques in the form of *timers*, *clock gating*, *supply shutdown*, and *operating system-based power management* (OSPM). Industrial standards have been facilitating the development of OSPM techniques. The *Advanced Configuration and Power Interface* (ACPI) standard is an OS-independent power management standard for personal computers (PC's).

B. Survey of Power-Aware Mobile Multimedia [191]

The three components of wireless handheld devices that account for most of the power consumption to run multimedia applications are processor, WNIC, and display. Much processing power is needed to perform a wide variety of tasks: signal processing in the codec unit for motion estimation and compensation, forward and inverse discrete cosine transfors, and quantization. Similarly, video encoding consumes much power. Communication cost depends not only on the amount of data to be downloaded/uploaded, but also on time-varying channel and rate of data transfer. Zhang et al. [191] have summarized the advances in video coding and delivery for power-aware mobile multimedia applications. Specifically, they focus on the following aspects of wireless multimedia: power-aware wireless video communication [204], [177], [84], [157], [203]; power-aware video coding [101], [102], [100]; and power-aware video delivery [206].

C. Survey of Energy Conservation in Sensor Networks [64]

III. Smart Batteries

Handheld devices use rechargeable electrochemical batteries. Their charging time is between 1.5–4 hours and they run for a few hours, though newer pocket personal computers run as long as 14 hours. For efficient and effective utilization of a battery, it is important to treat the battery as a measurable resource whose *attributes* are available to the operating system and applications on demand. A smart battery is a rechargeable battery augmented with additional sensing and SoC computation logic. *Fuel gauging* is a fundamental concept in the design of smart batteries. Intuitively, fuel gauging means how much charge is left in a battery and at what rate the battery is draining. The conceptual relationship of an application program with a smart battery has been illustrated in Fig. 2. The operating system interfaces with the smart battery logic to obtain the SoC value and makes it available to the application.

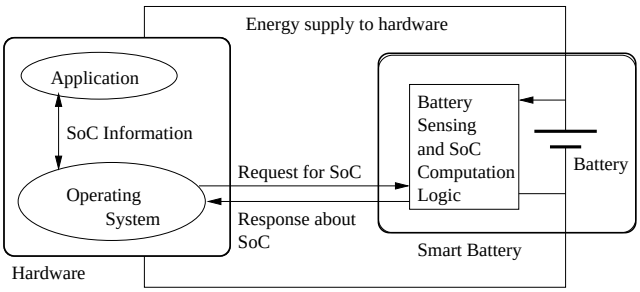


Fig. 2. Conceptual relationship between a smart battery and an application.

It is important for software designers to understand the discharge behavior of batteries, because the energy drawn from a battery is not always equivalent to the energy consumed in device circuits. In other words, a battery is a non-linear device, meaning that the remaining energy in a battery is *less than* the present amount of chemical energy in a battery minus the amount of electrical energy consumed by circuits. Rao, Vrudhula, and Rakhmotiv [1] have identified important battery attributes as follows:

- **Full design capacity:** It is the remaining capacity of a newly manufactured battery.
- **Full charge capacity:** It is the remaining capacity of a fully charged battery at the beginning of a discharge cycle.
- **Theoretical capacity:** It is the maximum amount of charge that can be extracted from a battery based on the amount of active materials it contains.
- **Standard capacity:** It is the amount of charge that can be extracted from a battery when discharged under standard load and temperature conditions.
- **Actual capacity:** It is the amount of charge a battery delivers under given load and temperature conditions.

Battery discharge behavior is affected by a number of factors, including the discharge rate, temperature, and the number of charge-recharge cycles. Those factors affect a battery as follows. First, battery capacity decreases as the discharge rate increases. Second, below room temperature, due to decrease in chemical activity and consequent increase in internal resistance, the full charge capacity decreases. At much higher temperature, the actual delivered capacity reduces too. Third, the popular, high density Lithium-Ion batteries lose a portion of their capacity with each discharge-charge cycle due to electrolyte decomposition. This loss in capacity is known as *capacity fading*. Therefore, accurate estimation of the SoC parameter of a batter is a difficult task

[2]. All the data in a typical smart battery are extrapolated from voltage, current, and temperature measurements [3], [4].

Predicting the lifetime of a battery is a difficult problem due to the fact that the amount of delivered charge, that is, the *actual capacity* of the battery, is a complex function of the physical and chemical characteristics of the battery and the time-varying load that is applied. During execution of user tasks, the hardware components draw a certain amount of current from the battery. This time-varying discharge current is referred to as a *load profile*, as shown in Fig. 3.

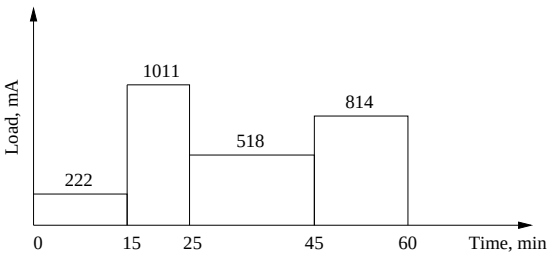


Fig. 3. An example of load profile [5].

The lifetime of a battery depends on the rate at which energy is consumed and the usage pattern of the battery. Continuous drawing of a high current results in an excessive drop of the remaining quantity of charge, and batteries do recover to a certain extent during intervals with no or very small currents. There is a need for a bettery model to describe the effects of the load profile and other operational parameters, such as temperature, on the state of the battery. Jongerden and Haverkort [11] have studied a number of battery models, and we summarize their key findings in the following.

A. Ideal Model: *Voltage* (*V*) and *charge capacity* (*C*) are two important properties of a battery. The charge capacity of a battery is expressed in ampere-hour (Ah). The quantity $V \times C$ is a measure of the energy stored in the battery. In the case of constant load current I, the lifetime (L) of a battery with capacity C is calculated as $L = C/I$. Due to non-linear effects, a simple approximation for the lifetime under constant load is: $L = a/I^b$, with $a > 0$ and $b > 1$. For variable load $i(t)$, battery lifetime is approximated as follows:

$$L = \frac{a}{((1/L)\int_0^L i(t)dt)^b}. \quad (1)$$

Equation 1 suggests that all load profiles with the same average current lead to the same battery lifetime. However, experimental results show that this is not the case, because the *recovery effect* has been ignored. Recovery effect means that the battery regains some of the "lost" charge capacity during its idle period.

B. Electrochemical Models: The electrochemical models are developed by considering the chemical processes taking place within a battery, and those describe the batteries in great detail. Consequently, those are the most accurate battery models. However, their highly detailed descriptions make the models complex and difficult to configure. A battery model in the form of six non-linear differential equations have been developed by Fuller et al. [18]. *Dualfoil* is a Fortran program that computes how the battery properties *voltage* and *current* change over time for the load profile set by the user. It is a complex task to use the model because the user has to set over 50 battery-related parameters [11], and to be able to do so one needs to have a detailed knowledge of the battery to be modeled.

C. Electrical-circuit Models: In these models, the electrical properties of the battery are modeled using PSpice circuits consisting of voltage sources, lookup tables, and linear passive elements, such as resistors and capacitors. At the core of the modesl are: a capacitor that represents the capacity of the battery, a circuit that discharges the battery, a voltage against SoC lookup table, and a resistor that represents the battery's internal resistance. Much experimental data are required to set the parameters of a circuit model, and the models are less accurate in predicting battery lifetime. Hageman [19] was the first to propose a circuit model.

D. Stochastic Models: Chiasserini and Rao [20], [21] have developed stochastic battery models with discrete-time Markov chains and applied them to communication systems [22], [23]. In the simplest model [20], the battery is modeled with a discrete-time Markov chain with $N + 1$ states, numbered 0 through N. The state number corresponds to the number of charge units available in the battery, and one charge unit corresponds to the amount of energy required to transmit a single packet. Therefore, N is the number of charge units directly available based on continuous use. In this model, every time step either a charge unit is consumed with probability $a_1 = q$ or recovery of one unit of charge takes place with probability $a_0 = 1 - q$, where the value of q is to be determined analytically. This model is too simplistic, because the rate of recovery is not constant during discharge, and, in general, the discharge current changes over time. A more accurate stochastic model has been developed in reference [23] and compared with the electrochemical model implemented in *Dualfoil*. The results of the stochastic model deviate from the electrochemical model by at most 4%. However, no numbers for the computed lifetimes have been reported in [23].

E. Kinetic Battery Model (KiBaM): The kinetic battery model uses a chemical kinetics process and was introduced by Manwell and McGowan [24]. The key idea in the model has been illustrated in Fig. 4, where the total battery charge capacity is distributed over two wells: the *available charge* well and the *bound charge* well. A fraction c of the total capacity is put in the available charge well ($y_1(t)$), and the remaining fraction $1 - c$ in the bound charge well ($y_2(t)$). Charge flows from the bound charge well to the available charge well through a "valve" with constant conductance k. The load current is denoted by $i(t)$ and is supplied from the available charge well. When a load is applied to the battery, the available charge reduces and the height difference $\delta = (h_2 - h_1)$ increases. Next, when the load is disconnected,

charge flows from the bound charge well to the available charge well until $\delta = 0$. Therfore, more charge is available when there is no load connected, and, in this way the model takes into account the recovery effect. The height difference δ is a measure of the replenishing capability of the battery, and for a constant load I, δ has been shown as a function of time t in Eq. (2).

$$\delta(t) = \frac{I}{c} \times \frac{1 - e^{-k't}}{k'}, \tag{2}$$

where $k' = k/c(1-c)$. Let the battery remain idle for a period t_i after a load I has been applied for a length of t_l. Then $\delta(t_i)$ evolves as given in Eq. (3).

$$\delta(t_i) = \frac{I}{c} \times \frac{e^{-k't_i}(1 - e^{-k't_l})}{k'}. \tag{3}$$

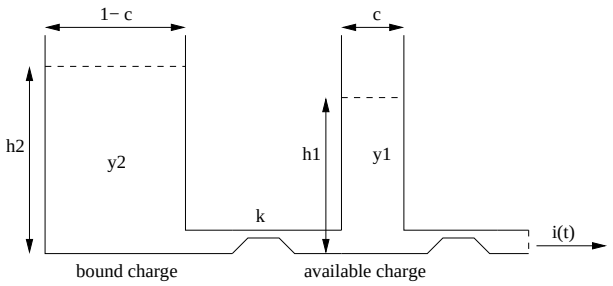


Fig. 4. Two-well Model of the KiBaM.

F. Diffusion Model: The diffusion model was developed by Rakhmatov et al. [5],[6] and it is based on the diffusion of the ions in the electrolyte, as explained in Fig. 5. In this model, the chemical processes at both electrodes are assumed identical, and only one of the electrodes is considered. For the fully charged battery, the concentration of the electro-active species is constant througout the electrolyte content (Fig. 5(a)). When current is drawn from the battery, the electrochemical reaction results in a reduction of the concentration of the electro-active species near the electrode (Fig. 5(b)). When the load is disconnected from the battery, the concentration of the species at the electrode will increase due to the diffusion, and the battery is said to recover (Fig. 5(c)) with a lower concentration. Finally, when the concentration at the electrode drops below a cutoff value, the chemical reaction can no longer be maintained and the battery is said to be fully discharged [11]. The details of their model in the form of differential equations can be found in references [5] and [6], and have been further explained in [11]. Rakhmatov and Vrudhula [5] have addressed the issues of energy management for battery powered embedded systems, composed of a processor, a voltage regulator, and a battery. They have studied the problem of task scheduling and selecting supply voltages, so that the resulting load profile yields maximum improvement in battery lifetime. Rakhmatov [14] has developed a more accurate battery model that relates the battery voltage to the load current, by capturing well-known nonlinear phenomena of capacity loss at high discharge rates, charge recovery, and capacity fading.

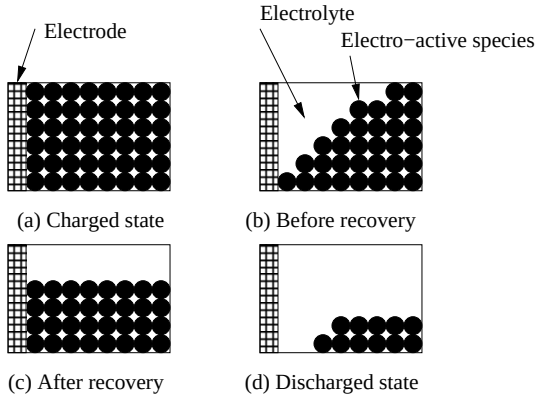


Fig. 5. Physical Picture of the Diffusion Model [5].

G. Other Models: Jackey, Plett and Klein [15] have reported an ESC (Enhanced Self-Correcting) cell model. They describe the construction and simulation of the battery model in the form of a *Simulink* diagram. They have proposed a semi-automated process for parameterizing a lithium polymer battery (LiPB) cell simulation model. Saslow [17] has proposed a simple battery discharge model for lead-acid cell that yields bahavior similar to what is observed in real life. Rong and Pedram [13] have proposed a high-level battery model to predict the remaining capacity of a battery considering temperature effect and the cycle aging effect. Their model relies on online current and voltage measurements, and the result has been validated by comparing it with *Dualfoil* simulation results. Zheng et al. [8] have proposed a physically justified iterative computing method, based on the model of Rakhmatov et al. [6], to illustrate the discharge, recovery, and charge process of Li/Li-Ion batteries. Rao et al. [7] have proposed a fast stochastic model as a 3-dimensional Markov process which is a stochastic extension of the KiBaM analytic model. Benini et al. [16] have proposed an abstract, discrete-time model for batteries, which is efficient to enable simulation-based battery lifetime estimation. They have described their model in the form of VHDL (Very High-level Description Language) code. The model has been validated by actual measurements with a number of different batteries, such as Lithium-Ion, Nickel-Cadmium, Alkaline, and Lead-Acid batteries.

Having studied a few battery models, Jongerden and Haverkort [11] have reached at the following conclusions:

- The analytic models, namely KiBaM and diffusion, are best suited to be used in battery lifetime predictions.
- The KiBaM model is in fact an approximation of the more complex diffusion model. Therefore, it is more suitable to use the KiBaM model.
- In predicting battery lifetime, it is more accurate to make use of stochastic workload models, so as to be able to capture the full range of different possible workload traces.
- A multi-battery system with an appropriate scheduling

scheme can be used to design dependable networked system [12].

It is important to understand that load profile has a significant impact on battery lifetime. Some load profiles may let a battery to recover from time to time, thereby maximizing the *actual capacity* of the battery, whereas some other load profiles may not let the battery to recover. Therefore, task scheduling is an important system-level activity to prolong battery lifetime. For communication-centric systems, such as handheld devices, transmission and reception of data packets consume much energy, thereby presenting opportunities to schedule those activites to improve battery lifetime. Chiasserini and Rao [22], [23] have identified the phenomena of charge recovery taking place under bursty or pulsed discharge conditions as a mechanism that can be exploited to enhance the actual capacity of the battery. They explore stochastic battery models to track charge recovery in conjunction with bursty discharge processes caused by bursty data traffic on the communication link of a device. They have shown improvement to actual capacity of the battery resulting from pulsed discharge driven by bursty stochastic load. Next, they have proposed discharge shaping techniques in order to be able to trade-off energy efficiency with delay [22].

Rakhmatov and Vrudhula [5] have given an analytical relationship among the load current $i(t)$, battery life-time L, and two experimentally determined battery parameters, α and β as follows:

$$\alpha = \int_0^L i(t)dt + \int_0^L i(t)\left(2\sum_{m=1}^{\infty} e^{-\beta^2 m^2 (L-t)}\right)dt, \quad (4)$$

where α is the battery's full charge capacity in coulombs, β denotes how quickly electroactive species (see Fig. 5(b)) are able to reach the electrode surface (the unit of β^2 is second^{-1}). Next, they have given a battery cost function $\sigma(t)$ as follows:

$$\sigma(t) = \int_0^t i(\tau)d\tau + \int_0^t i(\tau)\left(2\sum_{m=1}^{\infty} e^{-\beta^2 m^2 (t-\tau)}\right)d\tau. \quad (5)$$

The first term of the right hand side of Eq. (5) is the actual load or charge consumed in time t (denoted by $l(t)$), whereas the second term denotes the *unavailable charge* at time t (denoted by $u(t)$) due to load profile $l(t)$. Intuitively, unavailable charge is the amount of charge that cannot be extracted by a load from the battery. Now it is useful to make a distinction between the following two types of policies:

- *Battery-aware policy:* A battery-aware DPM policy tries to minimize the unavailable charge (the second term in Eq. 5.) at the end of a given workload.
- *Energy-aware policy:* An energy-aware DPM policy only minimizes the energy consumption (the first term in Eq. 5.)

Battery-aware DPM policies generally ignore the rest periods of loading a battery – and the rest periods can be user enforced, naturally occurring due to application characteristics, or due to finite load horizon [9]. Sometimes an energy-aware DPM policy leads to longer battery life than what a battery-aware DPM policy gives. For example, for both fine-grained ($<$ 10 ms) and coarse-grained ($>$ 30 min) tasks, battery-aware task scheduling methods to prolong battery life are ineffective.

In summary, software designers need to be aware of the battery characteristics in order to be able to schedule the major energy consuming tasks in such a way that the resulting load profile leads to the longest battery lifetime. The concept of a smart battery gives designers an opportunity to perform design trade-off, namely, performance versus application completion delay.

IV. Energy-efficient User Interface Design

Graphical User Interfaces (GUI) of desktop and laptop computers continue to be popular, and users expect the same – and even a higher – level of convenience from their small, mobile handheld devices, irrespective of resource constraints. The higher level of expectation is succinctly represented in the form of a mobile computing vision where devices can "see, hear, and feel" [43] for its user by identifying its dynamic context and supporting multimedia interactions [36]. As handheld devices become smaller in size, it is more difficult for users to interact with applications by their hands. Consequently, developers must consider the human factor and application controllability with flexibility. From the viewpoint of energy cost evaluation, a GUI can be broken down into two conceptual components, namely, *display* and *user interactions*, as shown in Fig. 6.

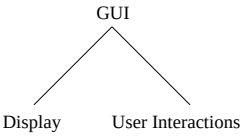


Fig. 6. Two conceptual components of a GUI.

A. Efficient Displays

Color and its manipulation are important considerations in designing interfaces for enjoyment and aesthetics in use [36]. Despite the limited energy available in a wireless handheld device, its display must have enough resolution and color depth to support for enjoyment and aesthetic use. A color TFT (Thin-Film Transistor) LCD (Liquid Crystal Display) display is common in handheld devices. An LCD display is composed of the following key components: an LCD panel, a frame buffer memory, an LCD controller, and a backlight inverter and lamp. A number of techniques have been proposed to minimize the energy consumption at the component level:

- brightness control of the backlight
- frame buffer compression
- dynamic control of color depth and refresh cycle
- dark window optimization.

Backlight Control: The dynamic luminance scaling (DLS) technique proposed by Chang, et al. [41] keeps the perceived

contrast of the image as close as possible to the original while achieving 20-80% power reduction from the backlight system for still images. The extended DLS technique of Shim, et al. [30] performs on-the-fly image compensation for still and moving images to achieve an average backlight power saving of 25%. Their technique is implemented in additional hardware integrated into the LCD controller. Cheng, et al. [29] propose a concurrent brightness and contrast scaling (CBCS) scheme to achieve a 3.7X backlight power saving with 10% of contrast distortion. They formulate and optimally solve the CBCS optimization problem with the objective of minimizing the fidelity and power metrics. Gatti, et al. [45] propose a technique to adaptively dim the backlight in response to changes in the ambient luminance. On the otherhand, the backlight adaptation scheme proposed by Pasricha et al. [42] is for streaming video, and it is executed on a *network proxy server*, obviating the need to modify the decoder on the device. They performed experiments on a Compaq iPAQ 3600 series mobile handheld device with TFT LCD screen and MPEG-1 video streams and showed energy saving of 9-60% in the handheld device's backlight. Cornea, Nicolau and Dutt [157] propose a *software annotation* based approach to enable a handheld device to control its backlight. Their technique is based on the following observation: while playing video, in many cases, the entire luminance range is not used, that is, there are many scenes in which only a few pixels are very bright. They suggest to increase the brightness of such images while simultaneously dimming the backlight. The technique involves analyzing video streams at the source (i.e. server) before they are transmitted to compute the required level of backlight and annotating the frames with that information. A drwaback of the approach is that reduced backlight luminance degrades picture quality. However in many scenes a small number of pixels with high luminance level may be sparsely distributed, thereby allowing clipping of some of those pixels without noticeable quality loss. One heuristic for pixel clipping is to randomly select a fixed percent of the very bright pixels to be clipped, which is a measure of the quality degradation metric. This approach can be viewed as providing trade-off between quality and energy. Both theoretical analysis and simulation studies show that up to 65% of backlight power can be saved with minimal or no visible quality degradation while running video applications on an iPAQ 5555.

Frame Buffer Compression: Even in power-hungry applications such as movie players, the display system dominates power consumption [32]. A frame buffer memory and buses are busy all the times due to the continuous sweep operations while an LCD panel is turned on. The energy consumption of the frame buffer and associated buses is directly proportional to the number of frame buffer accesses during the sweep operation. The number of frame buffer accesses is determined by the screen resolution, the sweep rate, and the color depth, which are generally constant. Therefore, the number of frame buffer accesses is constant in a typical architecture. Frame buffer compression reduces the number of frame buffer accesses and thus saves energy. Shim, Huang and Pedram [32] have shown that frame buffer compression reduces the display energy cost by about 50-66%, and the methodology saves about 10-15% of total power consumption in their prototype, where the LCD controller was implemented with an FPGA. An adaptive run-length-encoding (RLE) algorithm is presented in reference [32], whereas Shin, Cho and Chang [35] present a compression technique based on differential Hoffman coding, which gives better performance than the adaptive RLE algorithm. The compression technique based on differential Hoffman coding reduces the frame buffer activity by 52-90%.

Dynamic Control of Color Depth and Refresh Cycle: Choi, et al. [44] propose the idea of dynamic color depth control to reduce the energy cost of display. Color depth control involves pixel reorganization in the frame buffer, which enables half of the frame buffer memory devices to go into power-down mode at the cost of a decreased color depth. Moreover, reduced duty-ratio of refresh cycles lead to reduced energy consumption by the pixels on the TFT LCD panel.

Dark Window Optimization: It is not unusual for users to open multiple, overlapping windows while interacting with a device. However, not every window is of current interest to the user. Iyer et al. [33] propose the idea of modifying the windowing environment to allow changes to the brightness and color of areas of the screen that are not of current interest to the user. They performed a detailed analysis of display usage traces from 17 users, representing a few hundreds of hours of active usage. Their analysis revealed two important observations:

- On average, the window of focus uses only about 60% of the total screen area.
- In many cases the screen usage is associated with content that could have been equivalently displayed, with no apparent loss in visual quality, on much simpler lower power displays.

Based on the above two insights, they proposed energy-adaptive display systems that match energy use to the functionality required by the workload/user to obtain much energy saving. They proposed several dark windows optimizations that allow the windowing environment to change the brightness and color of portions of the screen that are not of interest to the user.

B. Efficient User Interactions

More than fifteen years back, when GUIs were in their nascent state, user interface code consisted of an average of 48% of the application code [49]. These days the user interface is almost always graphical, which only increases its fraction (i.e. share of code) and resource usage. GUIs are direct users of the display – and a display is one of the largest power consumers in a mobile device. Zhong and Jha [46] were the first to study the concept of energy-efficient and energy-aware GUI. They analyzed the energy cost of GUI from three perspectives, namely, hardware, operating system, and application, as explained in the following.

1) Hardware: A hardware perspective of energy consumption by a GUI has been illustrated in Figure 7. The energy cost of the LCD is dependent upon the display contents and their changes with time. At the pixel level, the energy cost of a pixel is a function of its color and the changes in the color as time passes. As a result of user interactions, GUI related interrupts are generated for the processor to produce new contents in the frame buffer. The LCD controller (LCDC) periodically refreshes the LCD with the contents of the frame buffer.

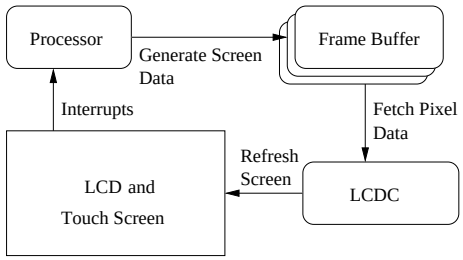


Fig. 7. A hardware perspective of GUI energy consumption [46].

2) Operating System: An OS perspective of energy consumption by a GUI has been illustrated in Figure 8. It is argued [196] that an operating system's kernel is the best location to implement software-based energy saving approaches (also, see [205]). The figure shows three software components: a GUI application, the GUI platform, and the OS. For a Unix-like OS the GUI platform is an X-server, and a GUI application interacts with the X-server in a client/server mode. User responses cause hardware interrupts to be generated for the OS to handle. As a part of interrupt handling, the OS causes the GUI application to execute its code. Execution of GUI application code makes it to communicate with the GUI server which may ask the OS to change the display screen.

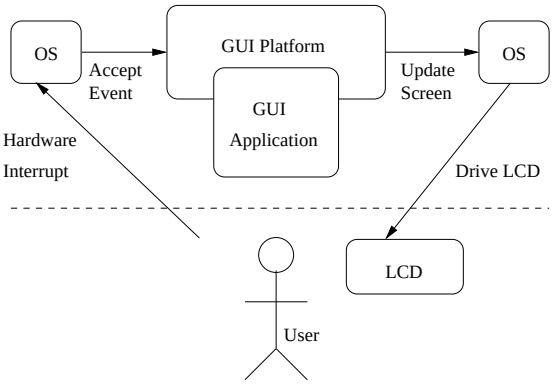


Fig. 8. An OS perspective of GUI energy consumption [46].

3) Application: An application (or, task) perspective of energy consumption by a GUI has been illustrated in Figure 9, which shows an interactive session of a user with a device. A *task* is defined to be a set of user actions and system responses, as illustrated in Figure 9, to achieve a meaningful goal, such as adding information on a contact or viewing a schedule [48]. The user thinks for a while, gives an input to the device, and waits until the device makes a response. After the device makes a response, the user looks at the response, thinks for a while before giving another input to the device, the device takes a while to process the response, and the cycle continues. While the user is in the think state, the processor and the communication module are idle, whereas the display is on. On the other hand, while the user is waiting, the device is computing a response and the display is on as well.

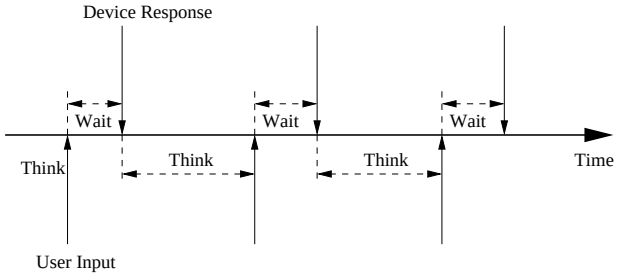


Fig. 9. An application perspective for GUI energy consumption.

Zhong and Jha [46] make the following recommendations to make GUI design energy-efficient and make a comparison with the traditional GUI design philosophies [50].

- *Accelerate user interaction:* Even an idle system consumes much energy so reducing the usage time for a task is an effective way of energy reduction. Programs and functions should be placed in the GUI so that a user can find them quickly.
- *Do something while waiting for user input:* Since idling costs much energy, the system can be proactive to accelerate user interaction by speculating the user input and get the result ready before the next input is provided. An example of speculation is the auto-completion feature of the virtual keyboard.
- *Minimize screen changes:* A screen change leads to changes in a subset of the pixels. Because it costs energy to change even a single pixel, it is recommended to minimize screen changes. Animation and window scrolling should be avoided.
- *Avoid or minimize text input:* Since text input is much slower, the amount of text input should be minimized. Instead, if the range of inputs is known beforehand, a list can be supplied to ask the user to choose from.
- *Reduce redundancy:* Features that do not accelerate usage should be avoided.

All the above recommendations, except the one to minimize screen changes, are also recommended by the traditional GUI design wisdom [50].

C. Understanding Human-Computer Interactions

Handheld devices spend much time in idle periods, waiting for user input. Displays have to be on in those idle periods – and it is known that displays are known to be one of the largest single power-consumers in mobile computers [46]. If a device assists its user in producing a quick response,

then the user's task will be completed by consuming less energy. According to the *model human processor* [51], the response time of a user is influenced by three fundamental processes, namely, *perception capacity*, *cognitive speed*, and *motor speed*. In the following, we briefly explain the three concepts while referring the reader to references [51], [46], [27], [53], [54], [26] for more details.

- *Perceptual Capacity:* A better visibility of the material being read has a positive impact on reading speed [52], and visibility depends on the font type and size, color scheme, contrast ratio, and luminance. GUIs with better color schemes and contrast ratios are easier to read.
- *Cognitive Speed:* If there are N distinct and equally possible choices, then the reaction time required to make a choice is given by the Hick-Hyman law as: *reaction time* = $a + b.log_2 N$, where a and b are constants. An interpretation of the law is that to accelerate the human cognitive process, a GUI should present as few choices as possible. Thus, the concept of *split menus* [25] is useful in realizing the above interpretation.
- *Motor Speed:* The motor speed, governed by the Fitts' Law [26], of human users positively impact their reaction time. User interactions often involve moving a control point (say, the mouse) from one position to another and activating a button at the destination position. The faster the motor speed of a user to move from the current position to the destination, the smaller is the idle time of the device. The time taken to move from the current position to the destination, denoted by T, is given by the Fitts' Law and is expressed as: $T = c_1 + c_2.log_2(\frac{D}{W} + 1)$, where c_1 and c_2 are experimentally determined constants, D is the distance between the two positions, and W is the width of the target. An interpretation of the law is that a GUI should utilize as much screen area as available for widgets to be rapidly hit [27].

D. Techniques for Designing Energy Efficient GUIs

Gong and Tarasewich [36] have discussed the characteristics and limitations of mobile device interfaces, especially compared to the desktop environment, and provided eight guidelines for designing user interfaces. Two of the guidelines are *design for top-down interactions* and *design for enjoyment*. Reading large amounts of text from mobile devices with small screens can require large amounts of scrolling and focused concentration. To reduce the amount of distraction, interactions, and potential information overload, it is suggested to present information through hierarchical mechanisms [40]. GUIs provide a way for users to interact with applications, and there are differences among GUIs. Before applying energy saving techniques to GUI design, it is useful to classify GUIs based on their primary interactions. Vallerio, Zhong and Jha [48] divide GUIs into three categories as follows:

- *Input-centric*
- *Content-centric*
- *Hybrid*.

The main task performed by a input-centric GUI is to obtain user input. For example, messaging and calculator applications have input-centric GUIs. The main task performed by an output-centric GUI is to deliver contents to the user. For example, map viewers and web browsers have content-centric GUIs. Some applications, such as text editors and configuration menus, require significant input and display components, thereby falling into the hybrid category. An input-centric GUI should be designed for ease of input, whereas an output-centric GUI should be designed for ease of browsing.

Specific techniques to reduce energy consumption by a GUI are divided into three categories:

- *Power reduction:* low-energy color scheme and reduced screen changes
- *Performance enhancement:* hot keys, user input caches, and content placement
- *Facilitators:* paged displays and quick buttons.

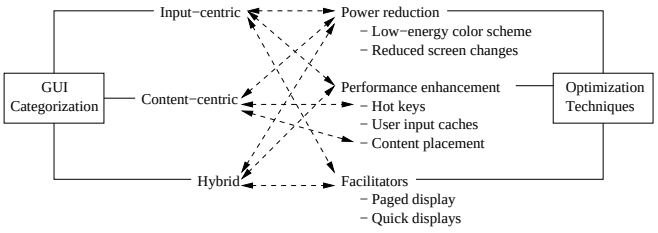
Two power reduction techniques are *low-energy color scheme* and *reduced screen changes*. Those are called power reduction techniques because they cause the display to consume less energy. For example, on TFT-LCD displays, a white portion consumes more energy than a black portion of equal size [46], [44]. For future organic-LED (OLED) based displays, the energy consumption of a display will be proportional to the number of on-pixels and their luminance [28]. Reduced screen changes mean reducing the total number of changes to the states (that is on, off, color, and luminance) of all the pixels. Screen changes can be reduced by reducing the usage of progress bar, scroll bar, and animation, because all of those features cost additional computation and cause the screen state to change.

Three performance enhancement techniques are *hot keys*, *user input caches*, and *content placement*. The performance enhancement techniques significantly reduce the total interaction time of the user for a given task, thereby reducing the total idle time of the device and the total energy cost. A hot key is a key combination (e.g. Alt + Ctrl + Delete) that the user can press to perform an action quickly. Hot keys increase user interaction speed on desktops, but not on mobile devices. However, the technique is viable through the use of quick button facilitators. User input caches speed up user interactions by storing the most recent or most frequent inputs by users. This technique is most useful when a small set of known inputs occurs frequently [48]. The autocompletion mechanism is a good example of the technique. For cellular phones, a good use of user input caches is to store commonly used phrases for text messaging. This will enable longer strings of text to be entered with fewer keystrokes, thereby increasing the speed of user interaction. Contents on a display page must be organized to reduce perception latency, motor latency, and cognition latency. The perception latency can be reduced by decreasing the amount of eye movement, which can be achieved, for example, by using the whole screen for up and down buttons, so the user does not have to focus on

the scroll bar on one side. To reduce motor latency, the Fitts' law explained before in this section is useful. Motor latency can be reduced by making frequently pressed buttons as large as possible and using the whole screen to control page scrolling. Cognitive latency can be reduced by decreasing the number of options from which users make selections, and split menus are a way to implement this concept.

There are two facilitator techniques, namely, *paged displays* and *quick buttons*. Facilitators do not reduce energy directly; rather, those enable other techniques to be used more effectively for energy efficiency [48]. For example, paged display increases the effective display size and allows the buttons to be larger, which can reduce energy consumption by increasing motor speed.

Not all the energy saving techniques can be applied to all three categories of GUI. The associations between GUI categories and energy saving techniques have been identified in Figure 10. Vallerio, Zhong and Jha [48] conducted a variety of experiments to show that the energy consumption of GUIs can be reduced by 16.4–45.2%.



Note: An association with Facilitators implies associations with all its components. Similarly, an association with Power reduction and Performance enhancement implies associations with all of their components.

Fig. 10. GUI categories and energy saving techniques.

A straightforward way of evaluating a user interface is to build a product and perform usability testing on the interface by having a large number of users actually use it. A similar approach has been take in [48] to evaluate the effectiveness of energy saving GUI techniques. Hyun, Kumazawa and Sato [37], [38] propose a new method to measure the positions of hands and the fingers when a user is manipulating button controls of a handheld device. They developed a hardware system for tracking each fingertip position by means of three strings and an artificial reality technique for monitoring the hand movements by tracing the hand in motion images captured by a camera.

E. Predicting User Interaction Time and Energy Cost

Modern user devices are characterized by their small form factors and heterogeneous user interface modalities, including pen-based touch screen, touch pads, trackballs, Gyroscopic mice, and hands-free interface, and there are continued developments in novel user interfaces [58]. Much heterogeneity of interaction modalities makes it difficult to predict and make decisions on energy consumption at design stages. Luo and Siewiorek [56] have presented a quantitative method, called Keystroke-Level Energy Model (KLEM), to predict the user time and system energy consumption to perform an interactive task. The KLEM method is an extension of the Keystroke-Level Model (KLM) [59] which is a cognitive modeling technique that predicts the time it takes a user to perform a given task with a given method on an interactive computer system. The *basic idea* of KLM is to list the sequence of keystroke-level actions that the user must perform to accomplish a task, and sum the time required by each action [57]. The KLM describes task execution in terms of four physical-motor operators (K, P, H, D), one mental operator (M), and one system response (R):

- **K:** pressing a key or a button
- **P:** pointing with the mouse to a target on the display
- **H:** moving hands to the home position on the keyboard or mouse
- **D:** drawing lines using the mouse
- **M:** mentally preparing for a task
- **R:** time-varying system response.

K, P, H, and D are determined by the actions necessary to accomplish the task. The KLM assumes that the first five operators take constant time for each occurrence, and provides a set of heuristic rules for placing M's in the sequence of Ks, Ps, Hs, and Ds, set by prior psychology and human-computer interface (HCI) research. The R operator must be estimated by the designer, and it includes only the time that the user must wait for the system after any M operator has completed. Designers can use *CogTool* [60], [62] to generate predictive cognitive models of skilled user performance by demonstrating their task on *storyboards* of the new design. The predictions made by CogTool are based on KLM, and are generated with a computational cognitive engine called ACT-R [61]. To make a prediction of interaction time, the following steps are followed by the designer:

- Create a *Storyboard* for the interface design. A storyboard contains a series of *Frames* that represent the changes of the display between user operations. A frame can be a sketch of the proposed user interface design.
- Identify the set of *Tasks* to be performed upon the interface.
- Demonstrate on the storyboard the steps a user would perform to accomplish a task. The demonstration is recorded to a script for CogTool to generate prediction.

Luo and Siewiorek [56] (also see [55]) propose a measurement based approach to obtain the energy profiles of KLEM operatrors by running a set of benchmarks on the target platforms. They define the energy consumption of a KLEM operator, the sum of the KLM operator energy, and the system activity energy it invoked. Specifically, let S be the set of power states of the system activity following an operator o , P_s be the power level of each state, and T_s be the time the system stays in state s. Then the system energy consumption triggered by the operator E_o is predicted using the following expression:

$$E_o = \sum_{s \in S} P_s T_s. \tag{6}$$

Let T_{task} be the total task time predicted by the CogTool, O be the sequence of KLEM operators in the task, T_o be the time of operator o, and P_i be the power level of the idle state of the device hardware. Then, the total system energy consumption during idle state E_{idle} is:

$$E_{idle} = P_i(T_{task} - \sum_{o \in O} T_o). \quad (7)$$

. Now, the total task energy E_{task} predicted is:

$$.E_{task} = \sum_{o \in O} E_o + E_{idle}. \quad (8)$$

.

KLEM can predict user time and energy consumption from storyboards of proposed user interactions with good accuracy. The predictions are within 13% of measured user time and system energy for two different popular handheld devices, namely, iPAQ and Tungsten. The time and energy for doing the same task in different user interaction modalities on the same platform (List Hardware Button vs. List Key) can vary by a factor of two to three.

F. Impacts of Human Factors on Energy Efficiency

As illustrated in Fig. 9, handheld devices spend much time and energy waiting for user responses, because of speed mismatch between computers and human users. In this subsection we explain the limits of user ineterfaces in becoming energy efficient primarily because of human factors [47]:

- *Sensory Perception-based limit:* A handheld device communicates with its user through the latter's sensory channels, namely, visual and auditory, and the device must spend some energy above some threshold levels for sensory perception. Details of those threshold energy levels as functions of viewing/hearing distance and wavelength of viewing light can be found in reference [47].
- *Input/Output-based limit:* As illustrated in Fig. 9, user response speed impacts the total time of task execution, thereby determining the energy cost. Typical values for user interactions, namely, speaking/listening/reading speeds, text entry speed, and stylus/touch-screen speeds can be easily measured. As an example, for reading printed English text, 250-300wpm (words per minute) is considered typical. Similarly, one can assume 0.5-1 second delay in GUI operations on handheld devices. The reading speeds and GUI operation delay translate to some minimum energy cost while the device is idle.

Zhong and Jha [47] have proposed the concept of a *wireless cache*, as illustrated in Fig. 11, to reduce the energy cost of waiting for user responses. The wireless cache is another piece of low-power communication device that acts as a bridge between the user and the handheld device. For example, the wireless cache can be implemented in the form of a wrist watch, and Bluetooth can be used to connect it with the handheld device. The handheld device can enter into a low-power mode while the wireless cache is waiting to receice user response.

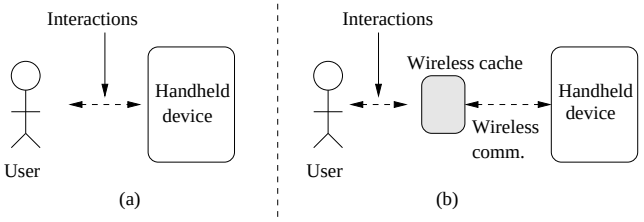


Fig. 11. (a) Standard user interactions with a handheld device and (b) user interactions via a wireless cache.

V. Sleep to Save Energy

Hardware subsystems, namely, processor and communication, spend much less energy in their sleep states than when they are active. In this section, we discuss the concepts of a micro-sleep state for processors and the wake-on-wireless for a handheld device are discussed.

A. μSleep – Another Sleep State

A state-transition diagram of a processor with three common states has been shown in Figure 12. If the operating system (OS) scheduler has an executable process, thread, or kernel code to execute, the processor remains in the normal *Running* state for as long as necessary. If the scheduler has nothing to execute at the moment, the processor moves to the *Processor Idle* state. When an interrupt occurs, the processor moves back to the *Running* state. The processor is explicitly put into the *Sleep* state by the user or by the executing code. For battery operated handheld devices it is useful to move the processor from *Running* to *Sleep* rather than to *Processor Idle* because of the significant difference in energy costs of the *Sleep* and *Processor Idle* states. The energy cost ratio of the *Sleep* (with LCD off) to the *Processor Idle* state can be 1:6.8 – 1:12.8 [106]. Therefore, it is useful to move the processor to the *Sleep* state as often as can be.

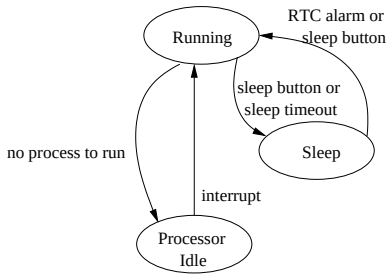


Fig. 12. State diagram of a processor without power saving mode.

To take further advantage of the significant difference in energy costs of the *Sleep* and the Processor Idle states, Brakmo et al. [106] have introduced the concept of a μ*Sleep* state so that under certain conditions the processor can be put in the μ*Sleep* state instead of the Processor Idle state, as illustrated in Figure 13. If there is no process to run, the processor is moved into the *OS Idle* state and the scheduler makes a decision to move the processor to the μ*Sleep* state or to the *Processor Idle* state. Entering sleep mode and

exiting it immediately after consumes a lot more energy than keeping the system idle for the same duration. Therefore, the transition from *OS Idle* to *μSleep* should be made if the scheduler determines that the processor can remain in the *μSleep* state for more than a *break_even* time and the devices allow the processor to sleep. A real-time clock alarm or an external event causes the processor to move back to the *Running* state. Their experiments have shown that *μSleep* can reduce energy consumption by more than 60% when the experimental Itsy pocket PC is lightly loaded.

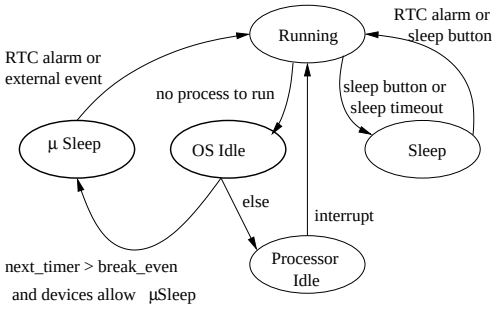


Fig. 13. State diagram of μSleep (RTC: Real-Time Clock).

B. Wake on Wireless

Wireless universal communicators (UCoM) fall into two general categories, namely, cell phones and PDAs. PDAs have the following advantages [108]: PDAs with IEEE.802.11b cards have high speed indoor-access to the Internet; PDAs are able to interoperate with a large number of devices because they use open standards, such as TCP/IP; a PDA-based phone affords the user access to a large number of mature, data and multimedia applications. In spite of the advantages of PDA-based phones over cellular phones, cellular phones outperform PDA-based phones in power consumption. A large part of the power drain in a PDA can be attributed to the WLAN card. A PDA connected to an access point (AP) drains its battery after a few hours even in the power-save mode, whereas cellular phones have standby lifetimes of a few days. For PDA-based phones to be successful as a UCoM, their battery lifetime must be comparable to that of cell phones. To extend the battery life of a UCoM device, Shis, Bahl and Sinclair [108] have proposed that when a UCoM device is not actively used, the device with its wireless network interface card be powered off. The device is powered on only when there is an incoming or outgoing call or when the user needs to use it. The above technique is called *wake-on-wireless*, and its goal is to almost eliminate the power consumed when a IEEE 802.11b-enabled device is idle. The technique is broadly explained as follows. A PDA communicates with an AP over two distinct radio interfaces, one for carrying normal data and one for carrying control information. The control channel is implemented over a low-power radio interface. When the device is not used, the device and the high-power wireless interface cards are shut down. To handle an incoming call, they use the low-power channel to send a wakeup message to the device. Once awake, the device accepts the call over its normal data interface.

The model of operation of the wake-on-wireless technique has been illustrated in Figure 14. The technique requires a second radio interface on each UCoM device, one or more UCoM Proxy, and one UCoM Server. The server is available on the Internet and is common to all UCoM client devices, whereas one Proxy serves all those clients who can communicate with it over their low-power, secondary radio interface. The Proxy too is connected to the Internet so that the Server and the Proxy can communicate. Figure 14 shows the three phases of operation of a device. In the first phase, that is initialization phase, all the client devices let the Server know of their presence and inactive devices, such as Alice, switch themselves off, except their secondary radio interfaces. The low-power radio module, called *MiniBrick*, of a client registers the device with the Proxy as shown in the figure. In the second phase, when an active client, say, Bob, wants to communicate with an off client Alice, the Server and Alice's Proxy collaborate to wake up Alice. In the third phase, Alice responds to Bob's invitation to establish a connection.

With an actual implementation and measurement-based evaluation of the technique, the authors showed that the standby lifetime of an iPAQ PDA can be increased by 115% over an unmodified iPAQ. With a normal talk of 25 minutes in a day, the above gain in standby time reduced to 40%. The main drawback of the technique is that it will require a large number of Proxy machines to be deployed, because of the short range of the secondary radio interface of each client device. A secondary drawback of the approach is the need to maintain logical associations between an UCoM Server and the numerous Proxy machines.

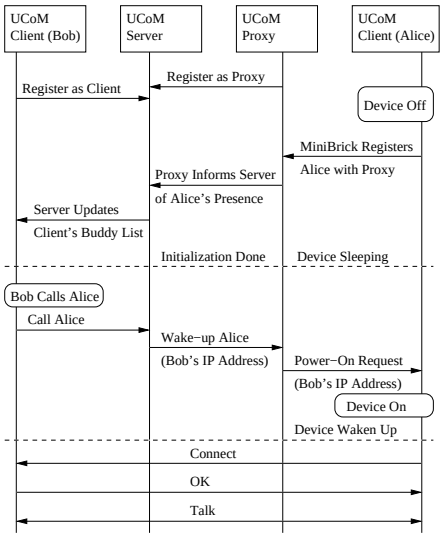


Fig. 14. Model of Operation of Wake-on-Wireless.

VI. Power Efficient Communication

A. Standard IEEE 802.11 Power Saving Mode (PSM)

The IEEE 802.11 [107] protocol has a built-in mechanism to allow user devices to perform dynamic power management of their wireless network interface (WNIC) cards. Such a card has five major states: *power-off*, *idle*, *transmit*, *receive*, and *sleep*. A WNIC card does not consume any energy in the *power-off* state. The *idle* state means that a WNIC card is powered on and it is ready to transmit or receive, and it consumes a significant amount of energy. The *transmit* and *receive* states are together known as the *active* state, and a WNIC card consumes the highest amount of energy in this state. Most of the circuitries of a WNIC card are turned off in the *sleep* state, thereby consuming much less energy than the *idle* state. The *sleep* state has been introduced in the protocol design to enable a device to perform dynamic power management as follows:

In a Wi-Fi hotspot, all communications occur via the Access Point (AP). In other words, the AP relays every frame to/from all user devices. Power saving is achieved by exploiting the central-controller role of the Access Point. That is, in the PCF (Point Coordination Function) mode of operation, each user device lets the AP know whether or not it utilizes the PSM. The AP buffers all frames addressed to user devices that have opted for the PSM. The general behaviors of user devices in PSM and the AP are summarized in the following.

- *Access Point:* While a user's WNIC is sleeping, data packets for the user are buffered by the AP. The AP periodically broadcasts *Beacon* frames, usually every 100 ms. A *Beacon* frame contains a *traffic indication map* (TIM) that indicates PSM user devices having at least one data frame buffered at the AP. The AP sends data to a user after the user device makes a request by means of a *PS-Poll* frame.
- *User's WNIC:* If the PSM feature of the card is enabled, the card sleeps for a *fixed period* upon the elapse of a *fixed timeout* since the last received packet. Upon expiration of a sleeping period, the card wakes up to listen for a *Beacon*. Therefore, PSM user devices are synchronized with the AP. If a user device is indicated in the TIM of a *Beacon*, the PSM user device sends a *PS-Poll* frame to the AP by means of the standard DCF (Distributed Coordination Function) mechanism to request the AP to send data. As illustrated in Figure 15, a user device with no data to send or receive sleeps between two *Beacon* frames.

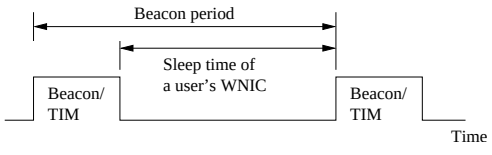


Fig. 15. General Idea of Beacon Period.

The power-saving feature is also supported in the DCF mode of operation. Time is divided into *beacon intervals*, similar to the PCF mode of operation. One purpose of the beacons is to synchronize all the user WNIC cards [107]. The power-saving feature of DCF requires that at the start of each beacon interval, each WNIC must stay awake for a fixed time interval, called *ATIM (Ad-hoc Traffic Indication Message) window*. All nodes are awake during the ATIM window. A node, say *A*, utilizes the ATIM window to advertize data frames pending transmission to other nodes, say *B*. The said advertisement is done in the form of an ATIM frame transmitted during the ATIM window by means of the standard CSMA/CA (Carrier Sense Multiple Access/ Collosion Avoidance) mechanism. The receiver of an ATIM frame (i.e. node *B*) responds with an ATIM-ACK frame and remains awake for the remaining duration of the beacon interval. If node *A* receives an ATIM-ACK frame, it remains awake for the remaining duration of the beacon interval. Node *A* sends data frames to *B* during the beacon interval, after the ATIM window.

The key characteristics of the MAC-level power management technique are as follows [80]:

- The technique does not completely switch off the card for additional power saving.
- The sleep duration is fixed, and it does not reflect variability in received data traffic.
- The maximum sleep time supported by implementations is usually less than one second, in order to reduce the risk of AP buffer saturation.

It is important to be aware of the drawbacks of an aggressive power saving policy, such as completely switching off the WNIC card.

- The policy will introduce longer delay in data traffic between a user device and an AP, because of the additional time needed for synchronization, authentication, and authorization.
- The energy cost of state transition from *switch-off* to *active* is high, and it must be compensated with a long off duration.
- An application-level protocol with knowledge of device (e.g. buffer size), data traffic (e.g. play rate and frame size), and network (e.g. link speed and core network delay) characteristics must be involved in making decisions to switch on and off a WNIC card.

B. Novel Power Saving Techniques for 802.11 Based Devices

Krashinsky and Balakrishnan [120] have shown by means of simulation studies that the IEEE 802.11 MAC's PSM can save around 90% of the energy spent by the WNIC at the cost of increased delay in web-page downloads. To cope with the delay problem, they propose a Bounded Slowdown protocol (BSD) in which a user device (i.e. mobile host) listens for the AP beacons with decreasing frequency during idle times so that it can be mostly sleeping during user *think times*. The time instants for waking up to listen for beacon frames are *statically* determined. As illustrated in Figure 16, think times are the time intervals when there is no network activity at the

MAC level. The BSD protocol trades off energy for lower additional delay. However, if the maximum tolerable delay is small, the protocol incurs more energy cost than the PSM. Qiao and Shin [121] have enhanced the BSD protocol – they call it Smart Power Saving Mode (SPSM) – by proposing a technique for *dynamically* estimating the time instants when a user device wakes up to listen for beacon frames. The Dynamic Beacon Period (DBP) algorithm of Nath, Anderson and Seshan [122] tries to reduce the delay by letting each user device select its own beacon interval and having the AP generate separate beacons for each user device. To counter the delay problem, Anand et al. [123] propose a Self-Tuning Power Management (STPM) protocol to exploit *hints* provided by network applications. The hints describe the near-future network activities of applications. One can view such hints as knowledge of the download times and think times as illustrated in Figure 16. The hints are used to save energy by managing the states of the WNIC. An application-specific energy optimization technique that enables a single (or few) burst of data transfer for an application over the WLAN interface has been proposed in reference [124]. The technique has been modified in reference [125] to estimate the idle periods of WINC cards so that it is applicable to a class of applications, rather than a single application.

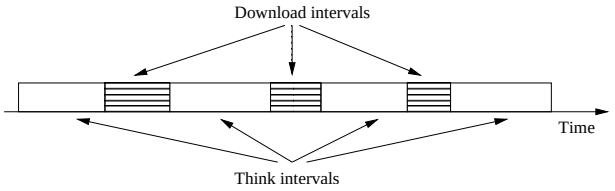


Fig. 16. Application-level Traffic Model.

The energy saving techniques in references [118], [116] and [119] use inactivity timers to decide when to switch off the WNIC. Stemm and Katz [118] use knowledge of application behavior to compute the timeout duration. Kravets and Krishnan [116] use an Indirect-TCP architecture [117] and buffer data at the AP while the user device is switched off. On the other hand, Yan et al. [119] estimate the interarrival times in a dynamic fashion to switch off the WNIC card.

Woesner et al. [110] have shown that the ATIM window size critically affects throughput and energy consumption, and a fixed ATIM window does not perform well in all situations. A too small ATIM window size may not allow a user device to successfully transmit ATIM frames to announce its buffered data frames, thereby degrading throughput. On the other hand, if the ATIM window is too large, there would be less time for actual data transfer. Moreover, large ATIM windows can also result in higher energy cost since all nodes remain awake during the ATIM window.

Jung and Vaidya [109], [115] propose a dynamic mechanism for choosing an optimal ATIM window size for energy saving in the DCF mode in an IBSS (Independent Basic Service Set) with the following characteristics:

- *Dynamic ATIM window adjustment:* Every user device independently chooses its own ATIM window size based on observed network conditions.
- *Longer sleep time:* A WNIC card enters the *sleep* state after completing the announced data transfers, rather than stay awake for the entire duration of a beacon interval.

We draw the attention of the reader to two important aspects of the above study: (i) evaluation of their technique is based on NS-2 simulations; and (ii) the proposed power saving mode does not work well with TCP traffic.

A number of proposals have been made to put a WNIC card to the sleep state after overhearing an RTS (Request-to-Send) or a CTS (Clear-to-Send) frame [111], [112], [113]. The technique in reference [111] needs a separate control channel to determine when to power off the second, data transfer channel and for how long. The technique in reference [113] is for sensor networks and it fragments long messages into many small fragments. The technique in [114] requires two time slots for beacons to be transmitted in each beacon interval: an ETS (Earlier Time Slot) beacon and an LTS (Late Time Slot) beacon. Presence of data traffic is announced in an ETS beacon, whereas, if no node has data to transmit, the LTS beacon is used to announce a sleeping period for all. This approach leaves less time for data transfer in a beacon interval, and heavily relies on all nodes having zero network traffic at the same time – an event whose probability is very low.

Anastasi, Conti, Gregori and Passarella [126] propose a Cross-layer Energy Manager (XEM) that dynamically tunes its energy-saving strategy based on the application behavior and network parameters. The general idea of the XEM technique has been illustrated in Figure 17 and summarized in the following:

- The technique assumes that the MAC level behavior can be described by an alternating sequence of network activity bursts and idle times, as illustrated in Figure 16. A burst represents data that are downloaded after the user device has sent a request to the fixed host on the Internet. It is further assumed that the first segment of a burst is sent by the user device, and it can be easily detected at the user device and identified by the request sent by the client application.
- Detection of the start of a think interval is performed in two different ways: *agent* based [124] and *timeout* based [125]. In the first approach, an *agent* on the user device spoofs the web traffic generated by the user [124]. For each web page, the agent is aware of the set of files composing the page. Once all the files have been downloaded, a think time is assumed to start. In the second *timeout* based approach, if no data is received by the user device after one TCP RTT (round-trip time), it is assumed that a user think time has started.
- During think intervals, XEM lets the device stay in the off mode.
- When an application-level request is detected, XEM switches the device to the standard PSM mode.

- XEM is able to achieve energy saving between 20% and 96% with respect to the standard PSM.

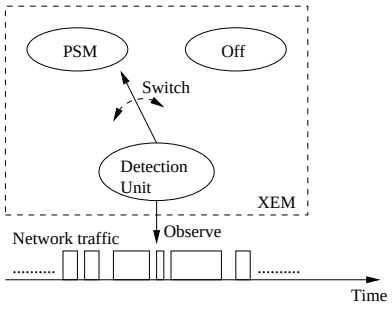


Fig. 17. The Conceptual Model of XEM [126].

C. MAC-level Download Scheduling by Access Points

In Section VI-A we discussed the power saving strategies in the IEEE 802.11 (WLAN) standard and some of its enhancements. Two key attributes of those strategies are as follows:

- Power saving in user stations (STA) is achieved due to the STAs themselves making decisions to put their WNIC cards into sleep or off modes, and the STAs are assisted by the AP.
- The STAs try to maximize power saving for themselves, without consideration for other STAs.

On the other hand, Lee, Rosenberg and Chong [136] propose an AP-centric power saving scheme:

- The AP is given a more central role in scheduling transmissions on the downlinks for more power saving.
- The scheme aims to minimize the total energy consumption due to communication of all the STAs, rather than the energy cost of individual devices.

A generic power management model (GPMM) has been introduced in [135] and further studied in [136]. They assume that the downlink and the uplink are separated, and the AP is in full control of the downlink. A STA can associate with the AP either in normal mode or in the GPMM mode. In the normal mode, an STA stays awake all the time. All the STAs are assumed to be GPMM enabled. A burst period is composed of Λ (= $L+1$) timeslots: one for transmitting the TIM and L for scheduling data packets. The *AP schedules* which STAs are going to receive their buffered packets in the coming BP using its own scheduling policy; next, the AP prepares a TIM and transmits a beacon packet when the time comes. It is further assumed that the STAs have to wake up at the beginning of each BP to listen for the TIM. If the bitmap for an STA is not set in the TIM, then the STA goes back to sleep immediately. Otherwise, the STA remains awake until the *last packet* scheduled for it in this BP is delivered. The MAC layer uses the concept of *last packet* to be able to transmit a long PDU (protocol data unit) from its upper layer by segmenting it into many smaller packets. The STA knows whether or not a packet is the last one by inspecting the MORE DATA bit field in the packet header. After receiving the last packet in this BP, the STA goes into sleep mode. It is assumed that the idle mode power consumption (P_{IDLE}) of an STA is the same as the receive mode power consumption (P_{RX}), and the sleep mode power consumption is negligible. It is further assumed that the energy consumption due to state change is negligible, and a timeslot is long enough for the AP to transmit one data packet. Data packets arrive at the AP continually, and packets arriving in a BP are scheduled for transmission in or after the next BP.

The scheduling problem has been formulated as an optimization problem: find an optimal scheduling discipline that minimizes the average total power consumption of all the STAs per BP. In other words:

$$\text{minimize } \bar{E} = \lim_{P\to\infty} \frac{1}{P} \sum_{q=1}^{P} \sum_{j=1}^{M} E_j^q, \tag{9}$$

where P is the number of BPs, M is the number of STAs, and E_j^q is the energy consumption of the jth STA during the qth BP. To optimize power consumption, they have developed two heuristic policies: (i) Longest Processing Time in selecting stations, Shortest Processing Time in scheduling packets (LPTSPT); and (ii) Dynamic Energy Efficient Semi-work-conserving (DEES). The performance metrics of interest (see Table I) are: the average energy consumption per time slot (denoted by E_{TS}), the average delay (denoted by $E[D]$), and the optimal value of the lengh of BP (denoted by Λ_{opt}), where M is the numbers of STAs, $\rho = \lambda.M$, λ is the packet arrival rate (number of packets/slot time), $Q = \lceil N/L \rceil$, and N is the total number of packets buffered at the AP.

Performance Metrics	LPTPTS	DEES
E_{TS}	$\frac{M}{\Lambda} + \frac{\rho(\rho.\Lambda+1)}{2}$	$\frac{M}{\Lambda} + \frac{\rho(\rho.\Lambda+Q)}{2.Q}$
$E[D]$	$\frac{(\rho+1)}{2}\Lambda + 2$	$\frac{(\rho+Q)}{2}\Lambda + 2$
Λ_{opt}	$\frac{\sqrt{2.M}}{\rho}$	$\frac{\sqrt{2.M.Q}}{\rho}$

TABLE I
MODELS OF PERFORMANCE METRICS OF THE GPMM MODEL [136].

The two policies have been analytically modeled and extensively simulated to study the energy saving and delay performance as functions of *offered load* (ρ), *length of BP* (Λ), and *number of stations* (M). They have shown how to dimension the optimal length of BP (Λ_{opt}).

Stine and Veciana [139] have proposed three alternative directory protocols that may be used by an AP to coordinate the transmission of data and the dozing of STAs. They try to optimize the power consumption of STAs by using *scheduling* and protocol parameter tuning. Their overall strategy is as follows: the AP first learns what traffic needs to be transmitted in a contention period (CP) and then directs the transmissions in a contention free period (CFP). They assume that the AP has learned of k transmissions in the CP, and focus on the CFP portion. They determine the control, scheduling, and error handling schemes that offer the best

energy conservation for the delivery of those k packets. The AP manages transmissions by broadcasting a directory – and the purpose of directories is two-fold: (i) manage who has access to the channel; and (ii) help STAs not involved in the transmissions to doze. A variety of directory structures have been proposed. At the very least, the directory identifies all transmitters that will be active during the subsequent CFP. At most, the directory will list the source and destination of all transmissions, their sizes, and the order in which they will take place. With this information, nodes can make informed decisions on when they may doze. The directories fall into two broad classes: *traffic indication maps* (TIM) and *traffic lists*. The AP assigns every STA a position in the bitmap when the STA first associates with the network. Depending on the number of bits used in each position of the TIM, a TIM may indicate which nodes will participate in a data exchange, whether they will transmit or receive, or even how many data exchanges they will make. The TIM does not normally indicate the order of transmissions or with whom the data will be exchanged. Consequently, the AP is required to direct each transmission during the CFP using a POLL packet. On the other hand, *traffic lists* provide all information about the CFP: each transmission has an entry in the list – and an entry specifies the source and the destination of each transmission and its duration. The tradeoff between TIMs and traffic lists are the length of the directory itself, the overhead to control transmissions, the amount of information that is available to help nodes to doze, and the ability of the protocol to adapt to transmission errors. Four kinds of traffic have been identified as follows:

- *Type I:* This is peer-to-peer traffic among STAs.
- *Type II:* This represents data traffic to and from a AP.
- *Type III:* This is downlink traffic from a AP.
- *Type IV:* This is uplink traffic to a AP.

They model the transmission time and energy consumption of the 1-bit TIM, multiple-bit TIM, and list protocols for Type III and IV traffic. Type II traffic is considered to be a combination of Type III and IV traffic. For Type I traffic, four scheduling algorithms have been presented. They observed that scheduling transmissions between the same nodes contiguously and ordering such transmissions shortest-processing-time-first (SPTF) results in good performance. The most critical feature that contributes to an access protocol's effectiveness in power saving is its ability to minimize the time it takes to inform STAs that they may doze. The mathematical model introduced by Chlamtac, Petrioli and Redi [144] to analyze centrally controlled access protocols has been utilized in [139].

The Energy Conserving MAC (EC-MAC) protocol presented in references [143], [142] also uses the AP to put the STAs to lower energy states. The EC-MAC protocol provides a *list* mechanism to schedule data transmissions between the base station and the STAs. After the list (i.e. the schedule) is transmitted, the STAs in the network are awake only during the periods that they exchange data. The mechanism performs efficiently when data to or from the same STA are sent in contiguous slots because of the time it takes for nodes to transition between states.

Havinga and Smit [137] have proposed a TDMA (time division multiple access) based energy efficient MAC protocol in which the AP schedules packet transmissions. The physical channel is divided into time slots, a data packet can span multiple slots, STAs can reserve slots for connections, and a *frame* consists of a certain number of slots. A *QoS manager* running on the AP receives transmission requests from the STAs. The QoS manager establishes, maintains, and releases wireless connections between the AP and the STAs, and it broadcasts a schedule so that the STAs know when they should transmit or receive. In the scheduling of the MAC protocol they apply the following two mechanisms to reduce the energy consumption:

- *Mobile grouping strategy:* This strategy means that an STA has a concatenated uplink and downlink phase, and that the transceiver will enter a low power mode for the remaining time of a frame.
- *Schedule traffic in bursts:* This strategy allows a transceiver to stay in a low-power mode for an extended period of time. Essentially, it enables frequent switching among the different modes of operation, namely, transmitting, receiving, idle, and sleep.

The QoS manager assigns bandwidth and determines the required error coding for individual connections. The same authors argue in reference [138] that QoS mechanisms are a tool to achieve energy efficiency in multimedia systems.

D. System Level Power Management of WNIC Cards

Simunic et al. [179] have studied a system-level joint power management and power control strategy for WLAN interface cards. For power management, they consider four states of WNIC cards: two active states, namely, *transmit* and *receive*, and two low power states, namely, *doze* and *off*. Doze mode consumes much less power than the active states, and it can be entered into and exited from with a small performance penalty (e.g. 0.8ms). When a WNIC card is off, it consumes no power, but it takes longer (e.g. 60ms) to enter into an active state. A power manager (PM) observes the workload (e.g. the input queue to the WNIC card) of the system and decides when to put the WNIC card in its off state. They define the concept of *renewal time* of the system as the last time the system entered the doze state. The problem of power management policy optimization is to determine the optimal distribution of random variable Γ which specifies when the transition from the doze state to off state should occur from the last *nenewal time*. Γ is assumed to take on discrete values in $[0, h, 2h, \ldots, jh, \ldots)$, where j is an index of time from reset state and h is a fraction of the time it takes for a system to make a transition from doze to off. Smaller the value of h, more accurate is the solution. By using renewal theory they formulate an optimization problem by means of liner programming to minimize the average number of jobs waiting in the input queue of the WNIC card. Thus, the goal of their power management optimization is

to minimize performance penalty under energy consumption constraint. The optimal solution is a table of probabilities $p(j)$, where $p(j)$ is the probability of transition from doze to off at time jh. Now the power manager works as follows:

- As soon as the system enters the doze mode (this occurs when the WNIC has no transmit or receive task), the PM starts keeping track of the elapsed time.
- The decision to turn off the WNIC card is evaluated at jh for which the table gives $p(j) > 0$. For each $p(j) > 0$, the PM generates a random number. If the number generated is less than $p(j)$, the WNIC card is switched off. Otherwise, the card remains in the doze state until the next evaluation or there is a need to make a transition to an active state.
- Once the card is switched off, it stays off until a need arises to turn on the card.

On the other hand, the WNIC card performs power control by adjusting its transmission power level with support from the AP. Essentially, the AP measures the received *signal to interference and noise ratio* (SIR) and *bit error rate* (BER) and calculates the transmit power level for each user at each time instant t. In summary, the proposal [179] saves energy in two additive ways: (i) by switching off the WNIC card from time to time and (ii) using less energy in its active state.

Simunic et al. [85] have presented the time-indexed semi-markov decision process (TISMDP) to turn off components to save energy for non-exponential user request interarrival times. Simulation results showed large saving in energy for three components: SmartBadge portable device and Sony Vaio laptop hard disk and WLAN card. Specifically, the TISMDP model resulted in three times less energy consumption compared to the default algorithm for the WLAN card.

E. Multi-radio Interfaces

Raghunathan et al. [190] have proposed the idea of a wireless personal server to readily store and access the data and appplications users carry with them through commonly found interfaces. A personal server is void of display and keyboard, but is equipped with multiple radio interfaces with varying capabilities. For example, a personal server can have a WiFi link and a Bluetooth link. On the one hand, the WiFi link is more efficient for bulk data transfers in terms of energy per bit transferred, bit it has a high idle power drain and state transition overhead. On the other hand, Bluetooth consumes low power, but it does not support high data rate. System designers can combine the advantages of the two radio technologies to support high data rate with a lower idle power consumption. Device discovery and connection establishment can be performed with Bluetooth, and data transfer can be done with WiFi.

We remind the reader that we are excluding the discussion of energy saving protocols for ad hoc networks (e.g. [140] [141]).

F. Power Management in WiMAX Subscriber Stations

The IEEE 802.16e standard, commonly known as WiMAX, is an emerging broadband wireless access system and defines both physical and MAC layers for combined fixed and mobile operations [127]. Since a mobile user device, called a Mobile Subscriber Station (MSS) in WiMAX parlance, is powered by battery, a *sleep* mode operation has been specified in the MAC protocol. The alternating *wake* and *sleep* modes of operation of an MSS has been illustrated in Fig. 18. The WiMAX standard defines three different types of Power Saving Classes (PSC) as follows:

- *PSC I:* Power saving for best effort and non-real-time variable bit-rate applications.
- *PSC II:* Power saving for real-time applications.
- *PSC III:* Power saving for multicast and management connections.

Note that the above power saving classes are essentially different sleep mode operations of MSSs. The first two sleep modes are about data traffic, whereas the third one is for management procedure. A class can be activated by either an MSS or the base station (BS) as follows:

- MSS initiated procedure: The MSS sends a *sleep request* (MOB-SLP-REQ) message to the BS with the following sleep parameters: the initial sleep window (T_{min}), the final sleep window (T_{max}), and listening window (L). The BS can accept, refuse, or change those parameters in a *sleep response* (MOB-SLP-RSP) message. After receiving a confirmation, the MSS can enter the sleep mode.
- BS initiated procedure: In this procedure, an *unsolicited sleep response* message is sent by the BS to the MSS, and the parameters cannot be re-negotiated by the MSS.

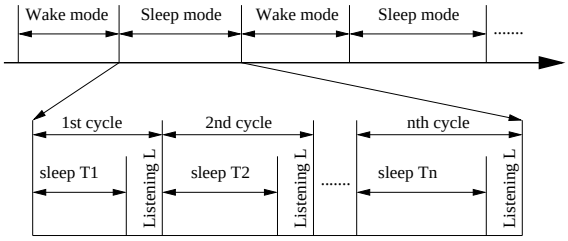
In order for the BS not to send data to an MS during its sleep windows, the BS is required to keep one *power saving sleep class*, which is uniquely identified by a sleep ID (SLPID). The idea of SLPID will be used later in this section to explain power saving. A class can be deactivated by either the BS or an MSS as follows:

- *Deactivation by MSS:* This is done when the MSS detects uplink traffic, and it is performed by sending a special MAC header containing information about the amount of backlogged data.
- *Deactivation by BS:* A BS deactivates a class only during a listening window of the MSS, and this is done by sending a positive *traffic indicator* (MOB-TRF-IND) message to the MSS during a listening window.

As illustrated in Fig. 18, an MSS comes out of its sleep mode from time to time to listen for the traffic indication message called MOB-TRF-IND and broadcasted from BS. The message indicates whether there is data addressed to the MSS. If MOB-TRF-IND indicates an absence of data for an MSS, then the MSS continues to sleep after the listening interval L. Otherwise, the MSS makes a transition to the wake mode. A sleep interval plus the following listening interval is known as a cycle. If an MSS continues to sleep

after not finding an indication of data in a MOB-TRF-IND message, it doubles the sleep duration in its following sleep cycle, and the maximum sleep duration saturates at T_{max}. In other words, the duration of sleep interval in the nth cycle is given in Eq. 10.

$$T_n = \begin{cases} T_{min}, & \text{if } n = 1 \\ \min(2^{n-1}.T_{min}, T_{max}), & \text{if } n > 1 \end{cases} \quad (10)$$



L: Fixed length listening window

Fig. 18. Wake and Sleep Modes in IEEE 802.16e.

In case there are incoming frames to a sleeping MSS, the device exits from the sleep mode in the *next* listening interval. On the other hand, if an outgoing frame reached the MAC layer during the sleep interval, the sleep mode is terminated immediately. Xiao [128] has modeled the expected time spent by an MSS in its sleep mode for Poisson arrival of frames at the base station for the MSS, but it does not consider frames to be uploaded to the base station. Zhang and Fujise [129] give an analytic model of power consumption for applications with both up stream and down stream data. Jang, Han and Choi [130] propose to make T_{min} and T_{max} adaptive to traffic types. For constant bit-rate (CBR) traffic, they propose to compute T_{min} based on average packet inter-arrival time (T_I). On the other hand, for FTP (File Transfer Protocol) traffic, they suggest to compute the values of T_{min} and T_{max} based on *queue empty* time. Kim, Kang and Choi [131] have modeled the *energy consumption* and *frame response delay* as functions of T_{min}, T_{max}, and T_I. Energy consumption is expressed in terms of listening intervals, sleep intervals, and energy costs per time units in those intervals. Frame response delay is the average elapsed time between the arrival of a frame at the BS (or, MSS) MAC layer and the completion of transmission of the frame to the MSS (or, BS) [134]. They show that a larger T_{min} consumes less energy but leads to longer frame response delay. When they set $T_{min} > T_I$, energy consumption does not reduce significantly, but frame response delay sharply increases. On the other hand, when T_{max} is a half of T_I or T_{max} is equal to T_I, energy consumption is not very high and frame response delay is moderate. Nga, Kim and Kang [133] give an algorithm to quickly find the optimal value of T_{max} in order to minimize the energy consumption with respect to a given frame response delay constraint. Han, Ming and Jeong [132] model energy consumption and frame response delay due to PSC I and PSC II. They conclude that small values of T_{min} (1 or 4 frames) can make PSC II reduce the response time greatly (up to 80% to 99%) at not much expense of the energy efficiency. Moreover, if T_I is small, very small values of T_{min} does not much sacrifice the energy efficiency while preserving a very low frame response time.

Anastasi et al. [134] have introduced the concept of *idle time* (T^{idle}) to reduce energy consumption and frame response delay. The T^{idle} parameter is used to detect *inactivity* by both the BS and MSS as follows. When a MAC Service Data Unit (MSDU) belonging to a connection of a SLPID (sleep ID) is received by the MAC layer, a timer is started equal to T^{idle} frames for that SLPID. On the other hand, if another instance of the same timer is running, then it is reset to the initial value T^{idle}. When the timer expires it is assumed that the connections of that SLPID will continue to remain idle for more time, and a power saving class is activated. Their simulation studies have shown that $T_{max} = 256$ gives a good trade-off between delay and energy saving. Also, T^{idle} should be set as low as possible since it affects only slightly the delay, but has a noticeable effect on energy saving.

VII. Proxy Assisted Energy Saving in Handheld Devices

A proxy is a request and content processing machine appearing between the server and the base station as illustrated in Figure 19. A proxy can be located anywhere in the communication network between the server and the base station, but it is generally located close to a base station and on the same access network as the base station. A proxy works as an intermediary between mobile hosts and streaming servers. Applications running on the mobile user device send requests for streaming media objects to the local proxy. In this section, we discuss how a number of different kinds of proxies enable handheld devices to save energy.

A. Energy Saving with a Transforming Proxy

Shenoy and Radkov [65] have proposed the concept of an object transformation proxy to save energy in handheld devices. The proxy services a request locally if the requested object is cached or fetches the object from the server if the cache does not have it [65]. In both the cases, the object is transformed by the proxy to make the device energy-efficient. To enable the proxy to transform the objects, the device sends three parameters to the proxy:

- the available energy budget for decoding ($\hat{E}_{dec}$),
- the available energy budget for reception ($\hat{E}_{bw}$), and
- the maximum spatial resolution ($\hat{R}$) that it can support.

Given the types of the CPU and the radio interface on the user device, the proxy is in a position to translate the decoding enegy budget into CPU time that the device is willing to spend and the reception energy budget into the size of the object that the device is willing to receive. The authors have proposed two techniques to reduce the energy consumption of a user device while it is downloading streaming objects.

- First, reduce the original size of an object to an appropriate size constrained by the amount of energy the user

device is willing to spend in decoding and receiving the object.

- Second, enable the user device to put its radio module into the power-saving, doze mode of operation by "intelligently" transmitting individual data packets so that the user device knows the arrival times of those data packets and goes into the doze mode.

Power-friendly Transformation: To reduce an original object to a desired size, they give two models for the decode energy cost (E_{dec}) and reception energy cost (E_{bw}) of video frames for a one-second interval at time t as follows:

$$E_{dec}(t, t+1) = c_1. \sum_{i=t.p}^{(t+1).p-1} D(f_i), t = 0, 1, 2, ... \quad (11)$$

$$E_{bw}(t, t+1) = c_2. \sum_{i=t.p}^{(t+1).p-1} S(f_i), t = 0, 1, 2, ... \quad (12)$$

where c_1 and c_2 are constants, $S(f_i)$ is the size of frame f_i, $D(f_i)$ is the decode time of f_i on the client device, and p is the play rate of frames. After determining the energy requirements E_{dec} and E_{bw} and the spatial resolution (R) of the original stream, the proxy transforms the stream if $E_{dec} > \hat{E}_{dec}$, $E_{bw} > \hat{E}_{bw}$, or $R > \hat{R}$. Frames are transformed in each one-second interval where the device-specified constraints are exceeded to generate a new stream of frames such that $c_1. \sum_{i=t.p}^{(t+1).p-1} D(f_i) \leq \hat{E}_{dec}$, $c_2. \sum_{i=t.p}^{(t+1).p-1} S(f_i) \leq \hat{E}_{bw}$, and spatial resolution is less than $\hat{R}$. A key challenge for a proxy to transform frames is to estimate the energy cost of decoding frames on the user device. They give an empirical model of the energy cost of frame decoding on a user device.

If an MPEG stream does not satisfy the client-specified parameters, the proxy transforms the stream to meet the energy consumption ($\hat{E}_{dec}, \hat{E}_{bw}$) and the spatial resolution $\hat{R}$. A key functionality of the proxy is to determine transcoding parameters to produce the transformed stream. The transcoding process essentially degrades the stream quality to meet the client needs in terms of energy cost. Video stream quality can be degraded along two dimensions: *spatial resolution* (i.e. picture size) and *chroma resolution* (i.e. picture quality). Shenoy and Radkov give an algorithm that searches various combinations of spatial resolutions and chroma resolutions by degrading along each dimension in each step until it finds an appropriate combination of the two. Essentially, for frames in each one-second interval, the algorithm uses the energy-cost models to examine the energy requirements for various combinations of spatial and chroma resolutions, picks one that satisfies the client needs, and invokes a transcoder. The algorithm produces frames with constant resolution but varying picture quality to make the video more pleasing to the eyes.

Power-friendly Transmission: This means assisting the user device in predicting packet reception, because a better prediction leads to greater energy saving by allowing the device to put its radio module in the doze mode for longer. The authors present two ways to make reception predictable:

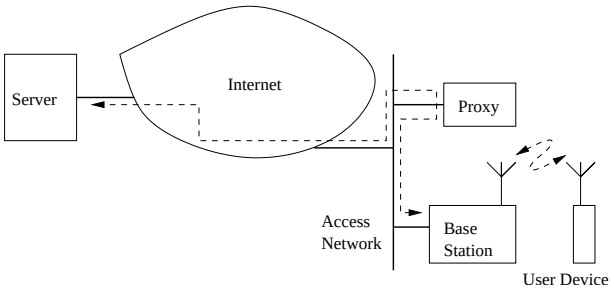


Fig. 19. A Model for Proxy-Assisted Energy Saving. The dotted lines represent flow of data and control packets.

- *Proxy-based smoothing:* Video streams employing compression formats such as MPEG can cause a variable bit rate (VBR) transmission from the base station, making it difficult for a user device to predict the next transmission. The proxy transforms the VBR data stream into a constant bit rate (CBR) stream within an interval I by setting the data rate to $\sum_{i \in I} S(f_i)/I$. With such an approach, a client device needs to estimate the arrival time of each packet and switch on its interface before the first bit of the packet arrives. One technique to predict the arrival time is to maintain a history of packet inter-arrival times and use the mean inter-arrival time (A) of the previous k packets to predict the arrival time of the next packet. If the previous packet was completely received at time t, then the predicted arrival of the next packet is $t + \gamma.A$, where $0 \leq \gamma \leq 1$. The value of γ represents how conservative or aggressive the device wants to be.
- *Proxy-assisted prediction:* In this approach, all frames within an interval I are transmitted in one burst, followed by a *control packet* specifying the start time of the next burst of frames. Specifically, if the proxy starts transmitting the next burst at time t, the user device goes into the active mode at $t+\gamma.d$, where d is the link delay between the proxy and the user device.

The proxy-assisted approach yields an idle uptime of 2-20%, whereas the proxy-based smoothing yields an idle uptime of 15-35% – smaller idle uptime means larger energy saving. Compared to an always-on radio module, the power-friendly transmission results in 65-98% energy saving in its radio module.

B. Backlight Control Based on Display Data Analysis

Cornea, Nicolau and Dutt [157] have proposed a technique for reducing display energy by analyzing video frames. Display energy is saved by simultaneously dimming the backlight while increasing the luminance of the image in order to compensate the lost brightness due to dimming the backlight. Brightness compensation can be performed for each individual pixel color (Red, Green, and Blue), and they propose to multiply all pixels in the image by a

constant amount to maintain the same perceived intensity (I): $I = \rho \times L \times Y$, where ρ is the transmittance of the LCD panel, and L and Y are the luminance of backlight and displayed images, respectively. Brightness compensation may lead to loss of picture quality due to pixel saturation. Therefore, picture quality degradation is defined in terms of the percentage of pixels in the high luminance range to be clipped (their luminance is represented by a lower value.) In many cases, one can safely allow clipping for some pixels without noticeable loss of video quality while saving much display energy. The technique is applied at the server or proxy as follows:

- Identify the scenes in a movie: a scene is a sequence of video frames in which the maximum luminance levels do not vary significantly.
- For each scene compute the required brightness level of backlight and annotate the information to the video stream.

Their results show that up to 65% of backlight energy can be saved through the aforementioned technique, with minimal or no visible quality degradation.

C. An HTTP-level Power Aware Web Proxy (PAWP)

Normally a user device puts its WLAN card in the sleep state (i.e. *doze* mode), if there is no communication activity for a predetermined interval of $T_{timeout}$, which is about 100 milliseconds. If data frames arrive at a user device from an access point with arbitrary time gaps between them, a user device cannot take full advantage of energy saving in a WLAN card in its doze mode. Rosu, Olsen, Narayanaswami and Luo [66] propose the concept of a power aware web proxy to enable a user device to put its WLAN card in doze mode for longer durations. A PAWP is located on the access network as illustrated in Fig. 20, and it works as follows:

A PAWP splits up a WLAN client-server TCP connection, buffers web pages requested by the device, and aggressively prefetches all the embedded objects in the pages. Consequently, data traffic between an AP and a user device is bursty in nature and the idle periods of communication between the two become longer, rather than fragmented, and the long idle periods result in longer sleep times and more energy saving for the WLAN card. The proxy works at the HTTP level, rather than the TCP level, allowing to exploit traffic information that is available only in the application layer.

Figure 20 shows the architectural details of a PAWP with four modules: client-side module, decision module, global state module, and server-side module. The client-side module accepts HTTP requests from user devices, and it builds a response if the requested objects have already been prefetched. If the requested objects have not been prefetched, the request is added to the data structure of the global state module. The server-side module handles interactions with remote servers: manages TCP connections, constructs HTTP requests, adds HTTP responses to the global state module, parses the responses to generate prefetch requests for the embedded objects and adds them to the global state. Every time the client- or server-side module changes the global state module, the decision module is activated. The decision module determines when a client request can be forwarded to the server, when a response can be sent to the use device, and when to reuse a TCP connection. The behavior of the decision module is controlled by an extensible set of rules.

A user's perception of page viewing quality may degrade because of data buffering at the PAWP. This is addressed by applying application-level rules in the decision module to shape user traffic: (i) data is released to the user device if it is available before the WLAN card switches to *power save* mode, which is computed as the last time a client request was received plus $T_{timeout}$; (ii) no object is delayed for more than a maximum delay, called $MaxDelay$; (iii) whenever more than $MinObjects$ number of objects are ready to be sent, they are forwarded to the client; and (iv) if enough data is buffered to justify the overhead of switching the WLAN to the active mode, data is forwarded to the access point. The authors performed a large number of experiments with popular sites, such as *CNN*, *NY Times*, *BBC*, *Amazon*, *eBay*, *Citibank*, and *American Express*, and reported energy saving for the WLAN cards on PDAs between 9% and 61%.

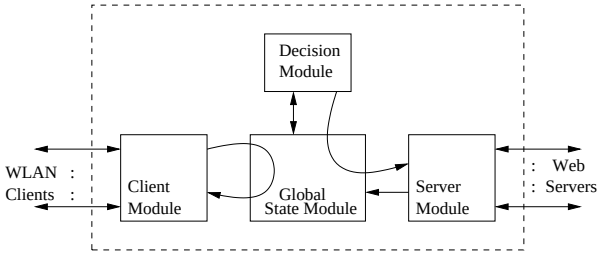


Fig. 20. Architecture of a Power Aware Web Proxy.

D. A Power Aware Streaming Proxy (PASP)

Rosu, Olsen, Luo, and Narayanaswami [67] have extended the PAxP (Power Aware x-Proxy) architecture to handle RTSP (Real Time Streaming Protocol)/RTP (Real Time Protocol) multimedia streams tunneled over HTTP. The extended proxy is called Power Aware Streaming Proxy (PASP). The tunneling process has been illustrated in Figure 21, where a client opens two TCP connections with a multimedia server by means of the *POST* and *GET* HTTP commands. The key difference between the PAWP and the PASP proxies is explained as follows:

- The PAWP architecture shapes traffic between an AP and a client device to be bursty, thereby making the lengths of *idle times of communication to be longer*, whereas the PASP proxy *transforms* the server-to-client multimedia stream to adapt it to client capabilities: the current WLAN link bandwidth, screen resolution, computing resources, and battery energy left.
- To hide the effects of stream transformation from the media server, PASP intercepts and alters the client-to-server stream tunneled on the POST connection.

Figure 22 shows the architecture of the PASP proxy with four modules: the client-side module, the server-side module, the stream transformation module, and the monitoring and control module. The first two modules are similar to the corresponding modules of the PAWP proxy. In addition, the client-side module collects information about the link bandwidth between the AP and the user device. The transformation module handles streams between client and server, in both ways. The monitoring and control module selects the stream transformations using available information. PASP performs the following transformations on the media stream:

- It forwards only the video layer that is most appropriate for the client rendering capabilities.
- It selectively drops the B- and P-video object planes while forwarding as many I-planes as it can.
- For streams with multiple video object planes (e.g. synthetic video), it may apply different transformations to different objects. For example, objects that are too small to be displayed on a mobile device screen may be deleted from the stream.
- Individual video packets can be dropped from a stream.

No further details about the performance of the PASP proxy have been reported by the authors.

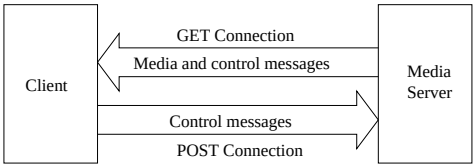


Fig. 21. HTTP Connections Used for Tunneling.

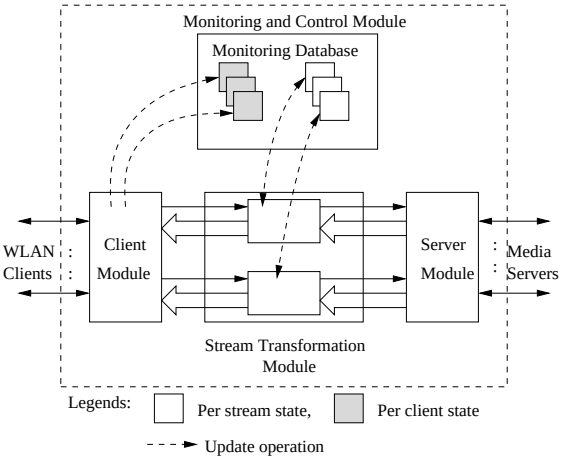


Fig. 22. PASP Architecture.

E. Proxy Caching for Streaming Media Delivery

A proxy is located near end-users between servers and users as illustrated in Fig. 19. If a data object is popular and it is expected to be downloaded by many users, it is useful to cache the data object on the proxy. Users receiving text-based objects can tolerate network delay and jitter, but jitter is annoying and it can drive clients away from streaming services. Partial caching approaches have been developed since full-object caching is not feasible for streaming objects [76]. A caching strategy becomes more useful if it can serve many users for a long period of time. In other words, if a cached object remains unaccessed for a long time, it simply takes up storage space with no use. The idea of partial caching is commonly adopted in caching streaming objects to tackle the storage problem due to full caching [77]. The typical partial caching techniques are categorized into two groups, also called dimensions, as follows [68]:

- *Viewing Time Domain:* The techniques falling in this category advocate the caching of the first few objects of a stream to reduce the client perceived startup latency. The subsequent objects are fetched while playing the initial objects. *Prefix caching*, *uniform segmenatation*, and *exponential segmentation* are typical techniques with a focus to reduce startup latency.
- *Quality Domain:* In this category, streams are organized in the form of multiple layers and multiple versions. Layered caching requires media objects to be layer encoded to enable the caching of the base layer, while enhancement layers are downloaded if there is network bandwidth. Multiple version caching statically caches different versions of a media object with different encoding rates, each one corresponding to a specific type of client network. After a proxy knows about the client type, the corresponding version is streamed.

Thus, a good caching strategy needs to be adaptive to dynamically changing popularity of objects. Chen et al. [68] have proposed two techniques to make a streaming proxy adaptive: (i) *adaptive and lazy segmentation*; and (ii) *active prefetching*. The first technique identifies and caches the most popular segments of each media object in the proxy, thereby maximizing cache utilization. The second technique proactively prefetches the uncached segments which clients are very likely to access. The caching techniques have not considered the energy saving aspect of user devices.

F. Streaming Audio Proxy

Anastasi et al. [70] have proposed a proxy architecture and an energy-efficient streaming protocol to minimize the energy consumption of the Wi-Fi interface on a device. The proxy service runs on access points (AP), as illustrated in Figure 23. The data path between the streaming server located on a fixed host on the wired network and the mobile device is split at the proxy on the AP. Real-time data is encapsulated in *Real Time Protocol* (RTP) packets [71] while control information are exchanged with the *Real Time Streaming Protocol* (RTSP) [72]. The TCP-Friendly Rate Control (TFRC) protocol [73] is used to control network congestion, because the underlying transport protocol UDP (User Datagram Protocol) does not perform congestion control. RTP and RTCP (*Real Time Control Protocol*) [74] packets carry TFRC information.

The *Real Time Power Saving* (*RT_PS*) protocol running on top of UDP and between the device and the AP is key to

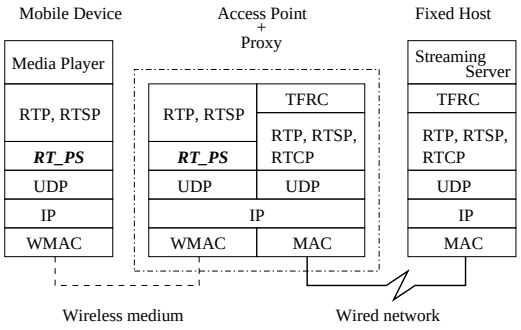


Fig. 23. Streaming Proxy Architecture.

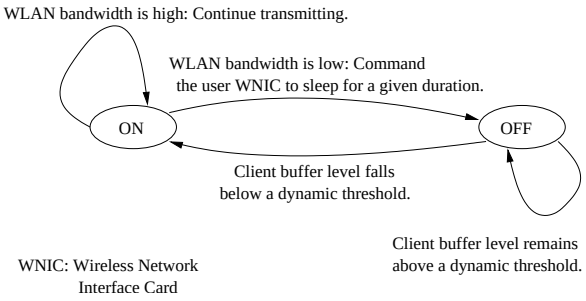


Fig. 24. The ON/OFF Model of the Streaming Proxy Architecture.

energy saving. The proxy component of the *RT_PS* protocol employs an ON/OFF model, as illustrated in Figure 24, and the protocol works as follows:

- Calculate the initial playback delay.
- Keep track of the WLAN bandwidth and the level of the user buffer.
- Keep transmitting in the ON state as long as the WLAN bandwidth is high. If the WLAN bandwidth becomes low, send a command to the client to switch off its wireless network interface card (WNIC) and move to OFF state. By monitoring the buffer level and knowing the play rate the proxy knows how long the WNIC can sleep.
- When the playback buffer falls below a threshold, make a transition to the ON state.

The client component of the *RS_PS* protocol works as follows:

- Request the proxy for an audio file. Let the proxy know the buffer size. Start playback after an initial delay.
- Let the proxy estimate the WLAN bandwidth currently available.
- Put the WNIC to sleep mode when asked to do so by the proxy, and bring it back to ON when the sleep time elapses.

The authors implemented a prototype of the proxy and performed experiments to show that the technique allows energy saving from 76-91% of the total consumption due to the network interface while maintaining a good user-level quality of service. The fraction of frames discarded by the proxy – those frames were estimated to arrive late for playback – or lost during transmission was below 5% of the total frames, and this fraction is considered acceptable because up to 20% loss in audio streaming is acceptable if loss concealment techniques are used.

VIII. Source-level Power Control

All of the power saving strategies discussed in Section VII share a common attribute, namely, proxy-assisted power saving. In other words, a proxy running between a client device and a media server assists the client device in moving its WNIC to sleep state. On the other hand, in this section, we summarize the energy saving strategies applied at the media sources on application servers.

A. Remote Power Control from Servers

Acquaviva, Simunic, Deolalikar and Roy [69], [78] have proposed a *proxyless* approach to enabling a client to move its WNIC to sleep state. In this approach, they exploit the server knowledge of the client workload, traffic conditions and feedback information from the client to perform traffic shaping over the wireless interface so that the client device can put its WNIC to sleep to save energy. The architecture of the server-assisted strategy has been illustrated in Figure 25. The architecture involves a power manager (PM), called *Client PM*, running on the client device and another power manager, called *Server PM*, running on the server. The two managers exchange power control information through a dedicated TCP connection, whereas the client application and the server application use a separate protocol/connection to communicate. The *Client PM* interfaces with the network driver of the client device, the local client application, and the remote *Server PM*. On the other side, the *Server PM* interfaces with the *Client PM* and the server application. The *Server PM* sends power control commands to the *Client PM* to: (i) switch off the WLAN; (ii) set the off time; (iii) enable the 802.11b power policy; (iv) set the 802.11b power management parameters, namely, the period between wake ups and the timeout before going back to doze mode. The *Server PM* performs the following tasks:

- *Client Adaptation:* The *Client PM* informs the *Server PM* about the application: input buffer size, the expected value and variance of the service rate (that is, rate at which the user device empties data from the buffer), WLAN card on/off transition time, and WLAN card status.
- *Traffic Adaptation:* The *Server PM* monitors both the wired and wireless traffic conditions. The server decides when to enable the WNIC power saving mode. For example, in light traffic conditions, the WNIC power saving might be used instead of a switch-off policy.
- *Traffic Shaping:* The *Server PM* schedules transmissions to the client in bursts in order to take advantage of the sleep state of WNIC. The burst size and delay are precomputed at the server. The delay should be large enough to almost empty the client input buffer, whereas the burst size should avoid overflow while keeping the buffer sufficiently filled. The idea is to maximize the off time of the WNIC and reduce the number of transitions

between on and off states.

The *Client PM* cooperates with the *Server PM* as follows:

- *Server Interface:* The *Client PM* obtains the *buffer size*, *depletion rate*, and *backlog* level from the application and forwards them to the *Server PM*. The device drivers report the transition time between off and on states.
- *Device Interface:* The *Client PM* responds to commands from the *Server PM* by calling the appropriate device driver functions to change the parameters of the 802.11b power manager, switch the WLAN on/off, and read the interface statistics such as signal-to-noise ratio.

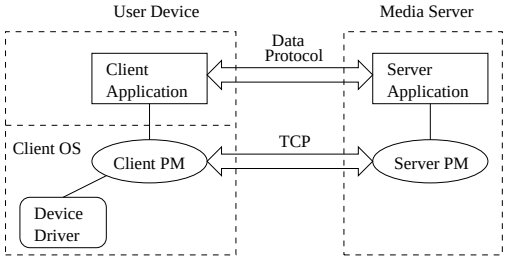


Fig. 25. Server Controlled Power Management.

The authors performed experiments by having the server transmit MPEG4 video data to a portable client. The RTSP protocol was used for session initiation and termination between the client and server. Media data units were carried using the RTP protocol over UDP (User Datagram Protocol). Two kinds of benchmarks were used to evaluate the energy saving strategy: (i) Benchmark 1 consisted of 12 bursts and ran at 15 frames/sec; (ii) Benchmark 2 had 402 bursts and ran at 30 frames/sec. Some frames were transmitted as single packets, whereas others required to be transmitted in multiple packets.

The delay between bursts, denoted by D_{burst}, is computed as follows:

$$D_{burst} = frame\_time.(\sum_{i=1}^{M} \frac{n_i}{i}), \tag{13}$$

where $frame\_time$ is the play time interval between successive frames, n_i is the number of frames consisting of i packets, and M is the maximum number of packets per frame. The first benchmark yielded a delay of 4 seconds between successive bursts. The strategy was tested on SmartBadge IV wearable device running a streaming video application. The strategy saved 67% of energy as compared to leaving the client WNIC always on, and 50% as compared to the 802.11b's built-in power saving policy.

B. Application-level Scheduling by Media Servers

Acquaviva, Lattanzi and Bogliolo [80] present two scheduling policies to exploit the information available at the application level to save energy. The two policies have been developed in the context of the system model illustrated in Figure 26. The elements of the system model and their key parameters are explained as follows:

- *Producer:* This represents the source of multimedia data and generates data packets. For each packet, three attributes are generated – randomly or from a trace: packet size in bytes, the frame it belongs to, and the total number of packets representing the same frame. Packets belonging to the same frame are produced in a burst.
- *Access Point:* An AP consists of two elements, namely, packet buffer and scheduler. The buffer is a FIFO queue with a tunable size. If the buffer is not full, the output packets of the producer are put into the buffer. Otherwise, the packets are dropped. The scheduler moves packets from its input queue to the buffer of the client device. If the client device is in power saving mode, then the scheduler remains idle, causing its incoming packets to be queued up in its buffer.
- *Wireless Channel:* The wireless channel between the AP and the user device is represented by channel latency, loss probability, and bandwidth.
- *Wireless Network Interface Card:* The WNIC card is assumed to have two operating modes: *always on* (ON) or *power-save protocol* (PSP). The details of the two modes have been represented as finite-state machines (FSM) in reference [80].
- *Power Manager (PM):* This models the actual implementation of the power management protocol of the IEEE 802.11b standard. The PM module generates events, namely, ShutDown and WakeUp, to notify the beginning and end of sleep periods.
- *User Device:* The user device has two key elements, namely, buffer and consumer. The buffer models the protocol stack buffer or the application buffer. The consumer represents a streaming application, and it decides how many packets to read within a frame period based on the information associated with incoming packets. Late frames and incomplete frames, caused by network delay or packet loss, are discarded by the consumer.

The two application-level dynamic power management (DPM) policies are as follows:

- *Closed-Loop DPM:* This policy has been illustrated in Figure 27. The client buffer is monitored with respect to two thresholds, namely, *high-water mark* (HWM) and *low-water mark* (LWM). The server continues refilling the client buffer. When the HWM threshold is reached, the client tells the server to stop sending and prepares to shut down the WNIC. It continues to receive packets until a timeout for the last packet occurs. When such a timeout occurs, it shuts down the WNIC to a radio-off state. The HWM value is made less than the buffer capacity to avoid buffer overflow caused by packet queuing at the AP. The video decoder on the client device consumes packets from the buffer until the level reaches the LWM, when the client wakes up the WNIC and informs the source to resume packet transmission. The effectiveness of the approach depends upon the distance between HWM and LWM and on

the transmission rate of the source. The source uses the LWM and HWM notifications to handle the packet needs of all the clients. LWM and HWM are estimated as follows:

$$LWM = (T_{oo} + T_c).\lambda_s \quad (14)$$

$$HWM = BS.(T_m + T_c).(\lambda_a - \lambda_s), \quad (15)$$

where T_{oo} is the off/on wakeup time of the WNIC, T_c is the cushion time used to avoid that the buffer empties completely because of network uncertainties, λ_s is the average packet consumption rate at the client and it is given by $\lambda_s = frame\_rate \times averageframesize$, BS is the client buffer size, T_m is the time needed for the client to send a HWM message to the source, and λ_a is the arrival rate of packets.

- *Open-Loop DPM:* In this approach, a client does not send the HWM and LWM notifications to the server. Rather, the server predicts when the client buffer will be empty by exploiting its knowledge of the workload. The server schedules the transmission to each client in bursts to create WNIC idle periods long enough to exploit the radio-off state of the WNIC. The client switches off the WNIC once the source has sent a burst that will keep the client busy until the next burst. The *burst size* and *idle time* between bursts are decided by the source in order to exploit the application buffer. The server transmits a burst of data preceded by a special control packet containing information about the burst size and the idle time D_{client}. The upper bound on burst size is given by the application buffer, whereas the idle time is bounded by the time taken to play a full buffer. It is important to note that the burst size is a trade-off parameter between energy saving and quality of application service. The key parameters are estimated by the source as follows:

$$\text{Burst size} = BS - N_c \quad (16)$$

$$D_{client} = T_{cons} - T_{oo}, \quad (17)$$

where N_c is a constant needed to avoid buffer overflow and T_{cons} is the time needed to consume the burst. Experiments conducted with three independent clients accessing the same MPEG4 server show that up to 75% energy saving in the WNIC card can be achieved without any frame miss.

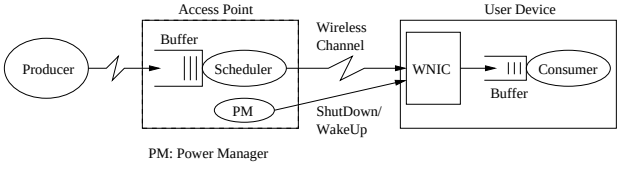


Fig. 26. Source-level Scheduling of Streaming Data in WLANs.

C. Application-level Power Control of WNIC Card

Bertozzi, Benini and Ricco [84] have studied an application-level, client-driven power control of WNIC cards,

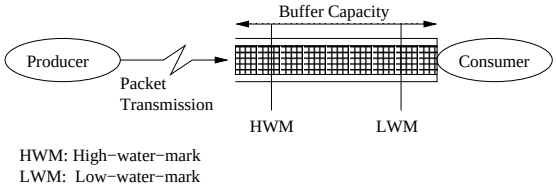


Fig. 27. Closed-loop Scheduling.

that bears some similarities with the *Closed-Loop DPM* explained in Section VIII-B. Their strategy can be applied while running an application that takes input from a buffer filled with data downloaded from a server. An example of such a scenario is playing a video stream. The application entity can ask the underlying OS to switch off the WNIC for a certain length of time while a streaming multimedia application is in progress. The WNIC is switched off until the playback buffer at the client is above a certain threshold. For this technique to be effective, several issues need to be addressed as follows:

- *Network connectivity:* By switching off a WNIC card the device loses network connectivity at the physical (PHY) layer level. Upon expiry of the switch off periuod, the device's PHY/MAC layer must perform channel scanning, authentication, and association with the AP once again.
- *Synchronization:* The PHY layer of a device loses time synchronization information when the WNIC card is powered off. When the card is powered on and an association is made with the AP, the PHY layer must reinitialize the synchronization information obtained from a beacon frame.
- *Power-up overhead:* Much energy is consumed in the form of a spike when a WNIC card is powered on. The power-on phase introduces some latency as well.

Switching on a WNIC card costs certain amount of energy so the number of switch on/off must be minimized, which is achieved by having a buffer of at least a certain size, determined by the play rate and download speed. The paper presents simulation results for energy saving as a function of buffer size and waiting time of media units in the buffer.

D. Source-level Resolution Control of Video Frames

Video applications in the forms of sports, news clips, movies, and personal videos, are becoming increasingly popular on PCs, laptops, and handheld devices. Transporting and playing streaming video are resource intensive tasks in terms of processing power, storage, and communication – and, consequently, energy. The experience of a digital video stream is characterized by three attributes [99]:

- *Temporal resolution:* The number of *frames per second* (FPS) is a measure of temporal resolution. A larger FPS value makes the scenes appear smoother. A standard value of FPS is 30.
- *Spatial resolution:* The total number of pixels, obtained by multiplying the row and column sizes (e.g. 352 $\times$

288), on a screen is a measure of spatial resolution.

- *Quantization parameter:* This denotes the quantization scale of the video under consideration, and it is represented by a certain number of bits (say, 32) to describe each pixel.

A dynamic technique for computing the optimal frame rate to deal with coding distortion has been presented by Takishima et al. [102]. Dynamic techniques have been proposed to compute frame rates to deal with fluctuations in available network bandwidth [103], [104], [105]. On the other hand, for a given frame rate and spatial resolution, a video encoder is able to control the picture quality by adjusting the quantization parameter to meet a specified bitrate [101], [100]. None of the above work considers the energy cost of handheld devices.

Lee, Lee and Shin [99] propose to control spatial resolution at the video source when the available bandwidth is low. Once a video encoder (i.e. the video source) or a proxy detects that the communication bandwidth is lower than a certain level, it starts encoding with reduced horizontal and vertical dimensions of the target device display. If a video decoder finds that the incoming video stream is spatially reduced, the display component resizes the decoded picture to meet the target dimensions. The authors have shown that use of reduced spatial resolution when the bandwidth is low allows handheld devices to provide the same, or even better, video quality with much lower energy consumption. Their approach did not have an advantage for spatial resolution above 176×144 because the energy required to resize a reduced frame is very high compared to the requirement of the decoding process. Resizing requires a lot of multiplication and addition operations, and the majority of those operations are floating-point calculations consuming much CPU time.

In Section IV-A, we explained how a server can assist a handheld device to dim its backlight [157]. The technique involves the source analyzing video streams and calculating the required backlight luminance levels while marginally sacrificing video quality.

IX. TCP Based Energy Saving

The Transmission Control Protocol (TCP) is widely used in web-based applications to reliably transport data between clients and servers. Flow control, congestion control, and error control by means of retransmissions are important functions performed by TCP. TCP's decisions to perform (re)transmissions cause packet transmissions at the MAC level, thereby influencing the energy cost of communication of a handheld device. As illustrated in Figure 28, we categorize the studies of TCP-related energy efficiency into two groups as follows:

- *Computational energy cost of TCP:* This represents the energy cost incurred due to executing the protocol details, such as congestion control, ACK generation, checksum computation, and round-trip time (RTT) measurement [90], [87], [86].
- *Controlling the WNIC for energy saving:* The WNIC card will remain idle if TCP stays inactive for a while, thereby giving an opportunity to put the card in a low-power state. Therefore, it is useful for the OS to monitor TCP activities to switch the states of the WNIC card [96].

A. Computational Energy Cost

Agrawal and Singh [90] have examined TCP's energy consumption behavior over wireless links and suggested improvements to save enegy. They performed experiments in which two laptops opened a TCP connection over a WLAN interface between the two, and one laptop continuously transmitted data over the TCP connection to the other. Both the laptops solely ran on battery, and their *lifetimes* and *volume* of data successfully transmitted were recorded. The general improvements are as follows: (i) larger MTUs (Maximum Transfer Units) lead to a larger volume of data transmitted, but cause the lifetime of the sender to be reduced, while increasing the lifetime of the receiver due to energy saving; (ii) smaller MTU sizes lead to longer lifetime with SACK (Selective ACK) on, because SACK option only increases efficiency when there are multiple losses in a single window, whereas for larger MTU size it is better to have the SACK option turned off because a window contains fewer number of packets of larger size and the probability of multiple losses is low; (iii) perform *header prediction* [91] *before* checksum testing to execute less code; and (iv) delayed ACK leads to fewer ACKs being transmitted, thereby saving energy at both the ends. Their experiments showed that the above improvements together can achieve energy efficiency of about 25%.

In a wireless network it is difficult to distinguish packet loss due to network congestion from packet loss due to bit error in the wireless medium. As a result, the classical concept of network congestion on fixed networks must be revisited for better TCP performance in wireless networks. There are six major variants of TCP for wirelss networks: *Tahoe, Reno, New-Reno, SACK, Vegas*, and *WestwoodNR* (see [87] for their salient fatures). Seddik-Ghaleb, Ghamri-Doudane and Senouci [87] have studied the effect of those TCP variants' congestion control algorithms on the following three performance metrics for different loss types that may occur in *ad hoc* network environments:

- *Energy* consumed in transmission, reception, forwarding, and retransmission. The energy cost is normalized to *per-bit* cost.
- Average *connection time* of TCP sessions between arbitrary pairs of nodes in an ad hoc network; and
- Average *goodput*.

The loss types were

- *BER* (bit-error rate) values of 5%, 10%, and 15%; and
- *lost link* scenarios, where a node is programmed to disappear for a while at a certain moment during the simulation to simulate *burst* errors.

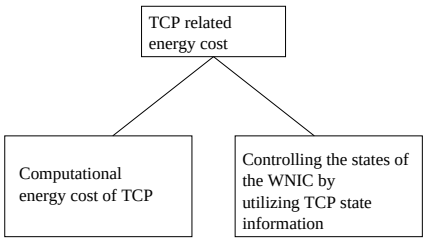
Their findings are summarized as follows:

- Congestion control algorithms allow for greater energy savings by backing off during error bursts.

- TCP New-Reno performs the best among the six variants.

In a recent paper [86], the same authors have reported the results of a measurement study of the computational energy cost of TCP New-Reno, Vegas, SACK, and Westwood in more general packet loss models, namely, *congestion*, *link loss*, *signal loss*, and *interference* with different ad hoc network routing protocols. They run TCP between two laptops connected over a wireless link. The *computational energy* cost of *sender* TCP is obtained by subtracting the measured communication energy cost of the WNIC card from the total energy cost of the CPU unit. They used a minimal Linux distribution and processes that were not relevant to TCP operations were disabled. Their idea was to ensure that the remaining energy consumption is mostly due to executing congestion control algorithms and timer adjustments. NS-2 [89] trace files were fed to the SEDLANE [88] system to emulate the loss and delay effects within their test-bed implementation. They observed that the computational energy cost of the Fast Retransmit/Fast Recovery phase is low compared to both of the Slow Start and Congestion Avoidance phases. TCP Vegas has almost no data loss compared to the other variants, because Vegas tries to avoid congestion by calculating and modifying its transmission parameters with each received ACK. This reliability is achieved at a higher computational energy cost than the other variants. On the other hand, TCP Westwood consumes less energy because it modifies its transmission parameters only after detecting data packet loss. TCP Westwood and New-Reno have almost the same energy cost even if the loss ratio is higher for New-Reno. Finally, though TCP SACK is capable of resending lost segments faster than New Reno, the cost of SACK processing and storage is high.

We are not summarizing their [86] observations about the impacts of traffic interference and link losses on energy cost in a mobile ad hoc network (MANET) while running routing protocols (e.g. AODV, DSDV, DSR, and OLSR). Bansal et al. [92], [93] have studied how the transmission energy efficiency decreases with an increase in the average number of hops for TCP sessions in a MANET. However, in this paper we stay away from further details of MANET related energy efficiency.



WNIC: Wireless Network Interface Card

Fig. 28. Two Ways of TCP Based Energy Saving.

Wang and Singh [95] have given a detailed break down of the computational energy cost of running the TCP on laptop and iPAQ, and developed approaches to reduce the energy cost of running TCP on handheld devices. They ran experiments on three different platforms: Toshiba laptop with a 650 MHz P3 Celeron processor running FreeBSD 4.2 and FreeBSD 5.0, and a Compaq iPAQ H360 PocketPC with a StrongARM processor running Linux 2.4.7. The network card was a 2.4GHz Lucent 802.11b WaveLAN "Silver" 11Mbps card. Power management was turned off both in the computers and the network card. The total power consumed by TCP was determined by subtracting the idle power consumption with power management turned off from the total power consumed during a TCP run. Their study showed that: (i) the *copy* operation between the OS kernel and the WNIC accounted for 60-70% of the energy cost for transmission/ reception; (ii) the copy operation between user space and kernel space accounted for about 15%; and (iii) TCP processing accounted for the remaining 15%. They further broke down the 15% TCP processing cost to show that checksum computation accounts for 20-30% of TCP processing cost. They have suggested the following techniques to reduce the computational energy cost of TCP:

- *Eliminating the user-to-kernel copy cost:* The user-to-kernel copy was eliminated by copying user data directly from the user buffer to the WNIC.
- *Reducing the kernel-to-WNIC copy cost:* This was achieved by: (i) maintaining TCP's send buffer on the WNIC; and (ii) maximizing the data transfer size from the kernel to the WNIC card. The first mechanism saves significant energy on all retransmissions since the segments will not need to be copied from the kernel space to WNIC. The second mechanism saves energy because of reduced number of interrupts and context switches as a result of copying data in larger chunks.

The above two techniques allowed the reduction of TCP processing cost by 20-30%.

Singh and Singh [94] have compared the energy costs of three implementations of TCP: Reno, Newreno, and SACK (Selective Acknowledgement). They performed experiments on a wirless testbed and measured the energy costs at the sender. The energy cost metric was calculated as the total energy cost of running TCP minus the idle energy cost. Experiments were performed under many network conditions: variable round-trip times, random loss, bursty loss, and packet reordering. Using total energy cost as the comparison metric, their conclusions are as follows: (i) in most cases, SACK outperforms Newreno which in turn outperforms Reno; and (ii) if a mobile device has a very low idle power consumption, then SACK fares worse than the other two implementations. However, given that devices consume significant power even in their idle states, the SACK is the best choice.

B. TCP-based State Control of WNIC Card

Bertozzi et al. [96] have observed that if an application reads data slowly from its local TCP buffer, then data flow at the TCP level remains idle for long periods of time. For

example, if a handheld device downloads a file and saves it into the local FLASH memory, then the application is forced to read data slowly from the TCP buffer because of the low speed FLASH. With one TCP connection on a device, an idle TCP implies that the lower protocol layers, including MAC, remain idle. Specifically, there are two relevant cases in which TCP experiences coarse-grained idleness: *full* buffer and *empty* buffer. The authors have proposed a technique to turn on/off the WNIC card of a handheld device based on the occupancy (or, utilization) of TCP buffer. The energy saving techniques are summarized as follows:

- *Full buffer:* When the receive buffer is full, the receiver TCP advertises a *zero* window size in an ACK segment, thereby preventing the sender from transmitting more segments. If the application reads data slowly from the TCP buffer, a small window size is not immediately advertised to avoid the *silly window syndrome*. The normal procedure followed by the receiver to handle the silly window syndrome is to test if it can accommodate one full-sized segment or half of the buffer is empty, whichever is less. To save energy, the OS puts the WNIC card to the sleep state if the buffer is full. The application continues to empty the buffer. When the receiver TCP advertizes a non-zero window size, as determined by the rules guiding the silly window syndrome, the OS wakes up the WNIC card, as illustrated in Figure 29.
- *Empty buffer:* A TCP buffer can remain empty for a long time if there is no application-level data transfer activity. In other words, an empty buffer is an indication of network inactivity, and the WNIC card can be put to the sleep state. When the OS recognizes data transfer activity or an application's read operation, it wakes up the WNIC card.

The authors have performed experiments on a testbed consisting of a Compaq iPAQ PDA running the Linux OS and a CISCO Aironet wireless adapter to validate the above proposed techniques. They reported energy savings ranging from 28% to 69% compared to the use of the state-of-the-art MAC layer power reduction techniques, with little impact on performance.

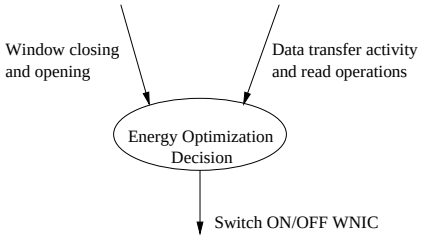


Fig. 29. TCP Optimization for Energy Saving [96].

C. Power-Saving Transport Protocol

Anastasi, Conti and Lapenna [82] have presented a power-saving network architecture (PSNA), as illustrated in Fig. 30, based on the concept of indirect TCP. The end-to-end transport-level connection between an application on a mobile host and a fixed server is split up into two transport connections: (i) a power-saving transport protocol (PS-TP) connection between the mobile host and the AP, and (ii) a standard TCP connection between the AP and the server. The interaction between the two TCP connections is asynchronous. A daemon on the AP acts as a proxy by transferring data between the two connections. The authors have reported up to 98% energy saving with the PSNA approach compared to the end-to-end, legacy TCP approach. The daemon uses a buffer to absorb temporary differences in the speed of the two connections; a speed difference may arise during the congestion recovery period for the connection between the AP and the fixed server. Since the PS-TP connection operates over one hop, it has been simplified as follows: (i) congestion control mechanisms are not necessary; (ii) efficiency of the error detection and recovery mechanisms are enhanced; (iii) it has been made connectionless. Thus, the PS-TP is a light weight transport protocol consuming less power to execute. For optimal power saving the PSNA strategy supports two high-level data transfer operations: *Read Access* and *Send Access*. The *Read Access* and the *Send Access* download and upload data from and to the fixed server, respectively, in an energy-efficient manner as follows:

- *Read Access:* A Read Access is performed in two steps. In the first step the mobile host delivers the read request to the end system by utilizing the services of the daemon on the AP. In the second step the mobile host retrieves the requested data from the AP by using the PS-TP protocol, without any interference from the congestion inside the Internet. The WNIC card of the mobile host is switched off between the above two steps for optimal energy saving. The time interval between the two steps is estimated by the AP based on the throughput of the TCP connection between the AP and the fixed host.
- *Send Access:* The mobile host requests the AP to open a TCP connection between the AP and the fixed host. Next, the mobile host sends a burst of data to the AP and gets an estimate of the residual time taken by the AP to send the data to the fixed host. If the residual time is more than zero, the mobile host puts its WNIC card to a low power state. After the expiry of the low-power state, the mobile host wakes up the WNIC card and starts sending another burst of data to the AP, and the process repeats.

In references [97] and [98], some early work on TCP-based energy saving can be found. Conti et al. [97] introduced the idea of using the indirect TCP concept to save energy by substituting a wireless link with a low-speed (i.e. 28-Kbps) serial line. However, the PSNA scheme is more developed than the one in reference [97]. A transport-level power management scheme for mobile devices has been proposed by Kravets and Krishnan [98]. Their idea is to shut down the WNIC card after an inactivity timer has expired and resuming it after a sleeping period. Inactivity timeouts and sleeping times are application dependent but fixed. On the other hand, the inactivity timeouts and sleep times are

dynamically adjusted to the traffic conditions in the PSNA scheme.

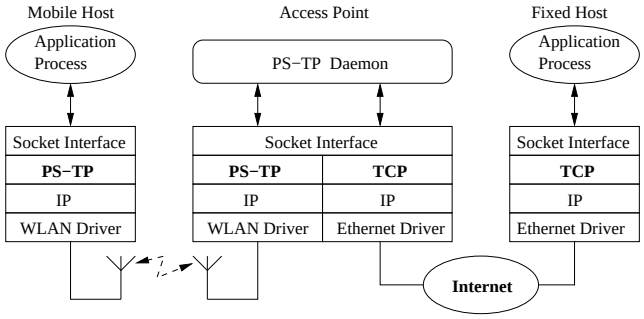


Fig. 30. Power-Saving Network Architecture [82].

Akella, Balan and Bansal [83] have proposed the idea of *multi-modal* protocols to conserve power by running in different modes of operations: power limited, bandwidth limited, and memory and/or computationally constrained. Specifically, they have considered various ways in which TCP can be redesigned to enable the *receiver* to conserve power, without requiring sender-side modifications. Their trick is to use TCP to make the sender transmit in a predictable manner.

- *Delayed ACK:* Delayed ACKs result in natural bunching of packets, and this causes the sender to send packets in bursts. For the receiver to get into the sleep state and conserve power, one way is to force the sender to transmit packets in bursts by using a delayed ACK mechanism. Here, computation of an appropriate delay value is important, because larger bursts may cause queue overflows at intermediate routers.
- Slow Start and Short Transfers: In the slow-start phase, packets arrive in bursts separated by time intervals of the order of the round trip time (RTT), and the receiver could sleep between bursts. They note that more than 70% of Internet traffic is HTTP web transfer based, and typical size of such a transfer is 8KB, which is about 7-8 TCP segments. In TCP based web browsing applications on handheld devices, most of the packets would fall in the slow-start phase throughout the duration of the transfer, and the received can sleep between bursts.

D. Lazy Scheduling Power Management

Bougard et al. [192] have studied the idea of *lazy scheduling*, originally introduced in [193] for a 1-hop wireless link, at the TCP level for energy efficieny. The idea of lazy scheduling is explained by using Shannon's equation for the maximum data rate (R) that can be reliably transmitted over a Gaussian channel characterized by a given bandwidth (W) and signa-to-noise ratio (S/N): $R = W \log_2(1 + S/N)$. The capacity of the wireless channel increases monotonically with the S/N ratio [194]. Therefore, reducing the required channel capacity allows decreasing the S/N ratio and therefore the transmit power. To what level the channel capacity can be reduced is determined by actual user requirements. Therefore, the key idea in lazy scheduling is to adapt the transmission rate to the actual user requirements, which are periodically approximated as the number of packets in the transmit queue. Lazy scheduling looks ahead, during a fixed time window (T_w), at the packet arrival from the upper layer of protocol. Let Δ_w be the actual queue backlog at the end of time window T_w. At the end of T_w, the transmit rate is given as follows:

$$\bar{R} = \frac{\Delta_w}{T_w} \tag{18}$$

To be able to control the average delay faced by packets, Bougard et al. [192] have proposed that the average rate of transmission be proportional to the queue backlog. In other words, $\bar{R}_i = K\Delta_i$, where Δ_i and $\bar{R}_i$ denote the queue backlog and average transmission rate at time instant i. For a linear queue, that is a queue that never becomes empty or full, the average packet delay $\bar{\delta} = 1/K$.

Padhye et al. [195] have shown that the steady-state bulk TCP transfer rate S is given by:

$$S = \frac{1}{RTT}\sqrt{\frac{3}{2p}}, \tag{19}$$

where RTT is the round trip time and p is the loss probability of a sent TCP packet. If MAC retransmissions are allowed, losses at the wireless link can be neglected, and p is dominated by the delays at the queue. Also, RTT is mainly determined by the queueing delays at the bottleneck link, which is most likely the wireless link. In other words, when the proportional lazy scheduling policy is used, both p and RTT are determined by the control parameter K. In the case of the normal lazy scheduling approach, delay is determined by T_w. From Eq. 19, an average rate constraint can be derived and propagated to the radio link control layer, which can use the information to decide the transmit rate and power level for a given channel attenuation for individual packets. Their simulation studies show an energy-scalability range up to 2.5 corresponding to a range of 2 in TCP throughput.

X. Upper-level Power Management

In this section, we discuss a few application-level energy saving schemes, namely, data compression, download scheduling, and task offloading.

1) Data Compression and Download Scheduling: Data compression is an attractive way for saving energy in a wireless handheld device, because the device remains active for a smaller duration to receive/transmit data over its wireless interface. For a download-centric application, data can be *compressed* by a proxy or the AP before download and *decompressed* by the user device before using it. While a user device saves energy to receive a smaller amount of data, it spends some energy to perform decompression. Thus, for a net saving in energy, the energy saving in the communication module must be more than the energy spent in decompression. Xu, Li, Wang and Ni [145] have performed experiments with three universal lossless compression schemes to make trade-off between the communication energy and the decompression overhead. The three compression schemes are the

gzip based on the Lempel-Zib coding algorithm [146], the *bzip2* based on the Burrows-Wheeler Transform [147], and the *compress* based on the Lempel-Zib-Welch software [148]. *Compression factor* is defines as the ratio of the input size to the output size, and the term *compression ratio* refers to the reciprocal of the compression factor.

The experiments have been performed on a *Compaq iPAQ 3650* with a 206 MHz Intel StrongARM SA1110 processor and 32 MB RAM, and the proxy server is a Dell Dimension 4100 with a 1 GHz P-III processor. The experiments involved a large number of files typical for use on handheld devices, such as web pages, documents, machine code, and media files. The two machines communicate through TCP socket calls and Lucent Orinoco IEEE 802.11b interface. The computing portion of the iPAQ is modeled as a two-state machine: *idle* and *busy*. On the other hand, the communication module has four states: *send, receive, idle*, and *sleep*. Their results are summarized as follows:

- If the raw file is large and compression factor is high, all the three schemes save energy. For small raw files, compression fares worse.
- If the compression factor is low, energy saving is marginal, because the communication cost is not reduced and decompression costs energy.
- Decompression efficiency has a significant impact on energy saving.

Handheld devices usually have simple hardware architectures with a single processor, and the processor is needed both for computation and for communication. The energy consumed while downloading consists of two parts: (i) the energy to receive and copy; and (ii) the energy consumed in the processor idle intervals between packet arrivals. Even with a full-speed downloading at 602 KB/s, the processor idle time is about 40% of the total receiving time. Consequently, about 30% of the total downloading energy is consumed due to processor idling. Xu et al. [145] have proposed to improve energy efficiency by filling the idle periods with decompression work: *decompress the i^{th} block of the data when downloading the $(i+1)^{th}$ block.*

Interleaving downloading and decompression can be achieved in the following ways:

- *OS kernel level:* Implement decompression in the TCP stack so that the kernel performs interleaving in an explicit manner.
- *User level:* Implement decompression as a user-process to perform decompression block by block. Since the receiving process is in the kernel interrupt handler, the receiving of the i^{th} block will interrupt the decompression of the previous blocks.

It is important to note that when the compression factor is low, even with interleaving, there is *net energy loss* in the range of 2% – 14%, compared to no compression. They have derived a formula to estimate the energy consumption for a given *file size* and a given *compression factor*, such that compression decision can be made adaptively.

2) Computation Offloading: The key idea in computation offloading is to migrate some computation-intensive tasks from the user device to a server on the network in order to save energy on the handheld device, as illustrated in Figure 31. Figure 31(a) illustrates the fact that an entire application executes on a handheld device, while Fig. 31(b) shows the application to have been partitioned and distributed between the handheld device and a server on the fixed network. The offloaded portion causes the device to spend less computation energy, whereas the communication energy of the device increases. If the offloaded task consumes more energy to execute on the handheld than the additional communication cost, then the strategy will lead to net energy saving [150], [182], [151], [152], [153]. Rudenko et al. [150] performed experiments on Dell Latitude XP portable computers running Linux OS and equipped with AT&T Wavelan card with a data rate of 2 Mbps. They chose three applications known to be fairly large and time-consuming: compilation of a large program, text formatting of a 200-page document using LaTeX, and Gaussian solution of a system of linear algebraic equations. A total of 220 experiments were made, consuming 900 hours. The results demonstrate that portable computers that execute their large tasks remotely can save significant amounts of battery power. They observed savings of up to 51%. Efficient ways of moving data from the portable computer to the server also save energy.

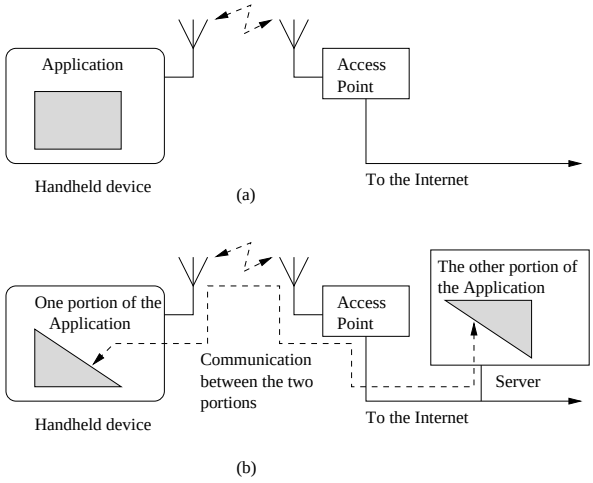


Fig. 31. (a) An Entire Application Runs on a User Handheld Device, (b) a Portion of the Application is Offloaded onto a Server.

Another approach to computation offloading is to partition an application into two halves such that one runs as a client on the user device and the other runs as a server on the network. Li, Wang and Xu [151] construct a *cost graph* for a given application program based on profiling information about computation time and data sharing at the level of procedure call. Next, the cost graph is partitioned to determine whether and how to offload the computation of a given program, and offloading units are defined at the level of procedure calls. The partitioning algorithm is based on the concept of *branch and bound*, and they have

given a pruning heuristic. Experiments have been performed on a Compaq iPAQ 3650 handheld device with an Intel StrongARM SA1110 206 MHz processor and 32 MB RAM, and a PIII 866 MHz Dell Dimension XPS. The two machines communicated over a Lucent Orinoco Wavelan interface with 11 Mbps speed. The test programs were multimedia benchmark programs taken from the *Mediabench* website. They reported that, even in an uncontrolled wireless LAN environment, the scheme resulted in significant energy saving for half of the benchmark programs.

Wang and Li [153] have studied a compiler-based tool to partition a program into a client-server distributed program such that the client code runs on the handheld device and the server code runs on the server. They use the concept of *abstract memory locations*, commonly used in pointer analysis techniques, and build a directed control-flow graph for the programs such that each vertex (v) is a *basic block* and an edge (v_i, v_j) represents the fact that v_j may be executed immediately after v_i. The partitioning problem has been formulated as an optimization problem that can be reduced to the *min-cut network flow problem* [154]. They give a heuristic method to deal with different program partitioning for different execution options. Experimental results show that the scheme can significantly improve the performance and energy consumption on handheld devices. Before Wang and Li [153], Kremer et al. [182] introduced a compilation framework for power management. They only considered the procedure calls in the main routine as the candidates for remote execution on the server and evaluated the benefit of remote execution of individual procedures separately. For procedures sharing common data, evaluation of remote execution of individual procedures may result in overcount of communication cost.

Many applications can be invoked under different execution options, input parameters, and data files, leading to different execution instances. Wang and Li [152] have considered the concept of *parametric program analysis* to deal with the problem of computation offloading of applications with different execution instances. They express computation workload and communication costs as functions of run-time parameters, and give an algorithm to find the optimal partitioning corresponding to different ranges of run-time parameters. At run-time, the transformed, distributed program self-schedules its tasks on either the mobile device or the server. The key ideas in reference [152] are as follows:

- A *task* is defined as a consecutive statement segment in a function that starts with a statement called *task header* and ends with a *task branch*, and there exist no task branches or task headers in the middle.
- A program execution is modeled as a TCFG (task control flow graph).
- At compile time, all the memory accesses are abstracted by *typed abstract memory locations*.
- Four kinds of costs have been identified: *computation cost*, *communication cost*, *task scheduling cost*, and *data registration cost*. The fourth cost is associated with the concept of typed abstract memory locations.
- Task scheduling between the user device and the server is formulated as a *min-cut network flow* problem.
- An iterative algorithm has been given to solve the parametric min-cut problem.
- Optimal partitioning decision is made at run time according to the run-time parameter values.

Experiments have been performed on a iPAQ 3970 Pocket PC client with a 400 MHz Intel XScale processor and a P4 2 GHz Dell Precision 340 desktop server. The test programs included *rawcaudio*, *rawdaudio*, *encode*, *decode*, *fft*, and *susan*, and those were executed with different options. For example, the *-3* option for *encode* uses G.723 coding for 24 Kbps bandwidth, whereas the *-4* option uses G.721 coding for 32 Kbps. Their results show that the optimal partitioning for computation offloading varies with different program execution instances.

Gitzenis and Bambos [180] have presented Markovian models to optimize jointly task uploading to a server and processor speed/power management. A task can be executed in three differeny scenarios as follows:

- *Local Execution:* This means the user's device executes the task locally, and it may apply DPM policies to trade-off execution speed with energy cost. Dynamic voltage/frequency scaling is an example of such trade-offs.
- *Remote Execution:* This means the task is executed on a remote server. Remote execution involves the following: (i) the device sends the task to the server over a wireless link; (ii) the server executes the task; and (iii) the handheld device downloads the results over a wireless link. Therefore, even with remote executions, a handheld device incurs some energy cost.
- *Combined Execution:* This means both the device and a remote server execute portions of the task.

The net gain or loss of energy with remote executions depends upon the cost of using the wireless link to send the task and receive the results, which depends on network connectivity, and the server load. Therefore, the concept of *stress* has been introduced to capture the quality of wireless links for the uploading and downloading actions, and the priority level that the server assigns to the task. Probability of bit-error (denoted by P_b) and probability of erroneous frame reception (denoted by P_e) have been used to describe the level of wireless link stress. The Markovian models are applicable to tasks with stochastically independent local and remote operations, that is, local execution, task upload, remote execution, and result download. Their optimization goal is to identify an optimal sequence of actions that minimizes the average energy cost accrued throughout the execution of the task, and the problem has been formulated in the context of Dynamic Programming. Further work needs to be done to develop concrete DPM policies for task offloading.

Martin et al. [10] have shown that energy saving techniques for communication and computation cannot be always studied in isolation. For example, one technique for

reducing communication energy usage is to perform more local processing, which obviously increases the amount of computation energy. Therefore, in the case of local versus remote processing, the communication and computation subsystems must be considered together. Reducing peak power can increase the battery lifetime of the system by increasing the actual capacity of the battery.

XI. Virtual Memory on the Network

In desktops and laptops with a high capacity storage memory (e.g. disk), the concept of virtual memory gives an application the impression of having an almost unlimited, contiguous random access memory (RAM) block. A virtual memory system makes the programming of large applications easier. Using remote idle memory as secondary store for networked systems is motivated by the observation that network speeds are getting faster more quickly than are disk speeds [172]. However, in handheld devices storage memory is small or absent due to weight, size, and power constraints. Therefore, it is useful to evaluate the efficacy of implementing a network-based virtual memory system for handheld devices. A key characteristic of a network-based virtual memory system is the need for additional energy cost incurred due to page swapping over the wireless link between a handheld device and a fixed network. Acquaviva, Lattanzi and Bogliolo [156] have studied the performance and energy cost of network swapping in comparison with swapping on local microdrives and flash memories using both synthetic and natural benchmarks. Experiments were conducted with Linux OS that performs page swapping as follows:

- Page swapping is activated in two situations: (i) when a kernel daemon, activated once per second, finds that the number of free pages has fallen below a given threshold, and (ii) when a memory request cannot be satisfied.
- In order to choose a page to be replaced, the process with the largest number of pages in RAM is selected.
- Among the page set of the selected process, the page to be replaced is chosen with an approximation of the *least recently used* (LRU) policy.

The experiments involved both *Compact Flash* (CF) and *hard disk* (mini and microdrives) on handheld devices and *network file system* (NFS) and *network block devices* (NBD) [81] on WLANs. User Datagram Protocol (UDP)-based Remote Procedure Calls (RPC) are used between the handheld device and the NFS system, whereas TCP-based communication is used between an NBD client running on the user device and an NBD server running on the WLAN. The user device consisted of an iPAQ 3600 handheld PC, equipped with a StrongARM SA1110 processor, 32 MBytes SDRAM and 16 MBytes of internal Flash. For local swap experiments, they used a 340 MBytes IBM microdrive and a 64 MByte Compaq Sundisk CF memory connected to the PCMCIA interface of the handheld device. The remote server was installed on an Athlon 4 mobile 1.2 GHz notebook. The WNICs used to provide wireless connectivity for the handheld device were a Lucent WaveLAN, a Cisco Aironet 350, and a Compaq WL110, while the AP connected to the swapping server was a Cisco 350 base station. The time bahevior of the supply current required by the PCMCIA card was monitored. In order to characterize the cost of a page swap, benchmark programs were developed to access data structures much larger than the available main memory without performing any computation on them. An example benchmark program is as follows: allocate and initialize a large matrix and, then, read by column in order to maximize the number of page faults.

The experiments measured the *time*, *energy*, and *power* required by each local and remote device to swap a page of 4 Kbytes, both for *read-only* and *write-back* operations. The dynamic power management support provided by the storage and communication elements are as follows:

- The CF has no *inactive* states, because its power consumption in wait mode is negligible.
- The microdisk's energy consumption in its wait state is about 65% of its read/write states. Therefore, it has a *sleep* state in which it consumes negligible energy.
- The WNIC cards of the handheld device has a low-power *doze* state and a zero-power off state. An IEEE 802.11b standard based WNIC provides MAC-level dynamic power management support that can be enabled via software.
- The NBD driver module switches on and off the WNIC card between swapping requests by making requests to the power manager module of an enhanced Linux.

It is important to note that the effectiveness of any DPM strategy is strongly dependent upon the workload characterization [156]: the higher the burstiness of the workload, the longer the idle periods, and consequently, the higher is the energy saving.

XII. Programming and Compilation Techniques

The potential for energy reduction through modification of software and compilation has been studied since the mid 90s [186], [161], [184]. Tiwari et al. [161] have proposed a *measurement* based *instruction* level power analysis technique that makes it feasible to effectively analyze the energy consumption of software. The concept of a set of *base costs* of instructions plays a key role in the aforementioned studies. The base cost of an instruction is the energy cost associated with the processing needed to execute the instruction. A number of energy reduction techniques have been proposed in the literature founded in the concept of base costs [186], [161], [184], [189]:

- **Reducing Memory Accesses:** Different machine instructions consume different quantities of energy, and instructions with memory operands have very high energy costs compared to instructions with register operands. Reduction in the number of memory operands can be achieved by adopting suitable compiler policies, e.g. saving the least amount of context during function call, and through better utilization of registers. Use of register operands reduces energy further because it can

also lead to shorter running time due to elimination of potential cache misses.

- **Energy Cost Driven Code Generation:** Here the idea is to select instructions based on their energy costs instead of the traditional metrics of code size or running time.
- **Instruction Reordering for Low Power:** In some applications, namely, signal processing, instruction reordering can lead to significant (e.g. 14%) saving in energy [185].
- **Instruction Packing and Dual Memory Loads:** Digital signal processing applications have a special architectural feature that allows certain pairs of instructions to be packed into a single instruction. The packed instruction executes in one cycle instead of two. The use of packing leads to 26–47% energy reductions.

Xian et al. [183] have proposed a programming environment by means of an architecture and run-time energy characterization, as illustrated in Figure 32. The framework allows applications and OS to collaborate for energy saving by exploiting the application knowledge of programmers. Programmers provide the OS with the values of application-level runtime conditions (*SpecifyCondVal()* call), querying the energy estimates of the options to decide what to choose (*QueryEnergy()* call), marking the start (*StartOp()* call), and stop (*StopOp()* call), of each option, and revising the options that do not save energy. An energy-aware program identifies different ways, called *options*, to achieve desired functionalities. The responsibilities of the OS include: measuring the energy consumptions of options, obtaining the values of the runtime conditions at OS level, and integrating all runtime conditions to estimate the energy consumption. The framework of Fig. 32 has two parts: (i) an API library that can be linked with programs; and (ii) a kernel module in the OS to support the functionalities defined by the API. The *Data Recording* submodule records energy consumption data, whereas the *Energy Prediction* submodule predicts the energy consumption of options.

The energy prediction submodule uses power parameters of devices and the kernel information about tasks to determine the energy consumption of individual programs. They use recorded data for energy cost prediction. Energy cost and runtime conditions are recorded in five tables, corresponding to CPU, disk read, disk write, network send, and network receive. An example of a runtime condition about data reading from a local disk is: *the data are not in memory, the disk is sleeping, and the bytes to read is 50 Kbytes.* The prediction algorithm is viewed as a *decision tree*. The leaf nodes represent the energy costs of the runtime conditions represented by the paths from the root up to the leaves. They have developed a Linux-based prototype of their framework to demonstrate the proof of concept.

Flinn et al. [181] have proposed a software architecture, called *Odyssey*, for implementing collaborative relationship between applications and the operating system with the objective of enabling an application to tradeoff its quality for

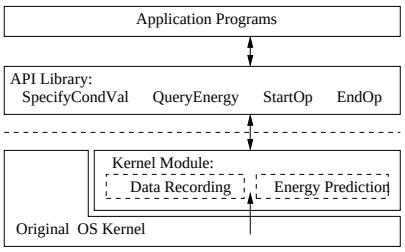


Fig. 32. An Architecture for Energy-aware Programming Environment. Arrows show information flow [183].

battery life. Odyssey captures the notion of data degradation through an attribute called *fidelity* that defines the degree to which data presented to a user matches the reference copy at a server. The architecture monitors energy supply and demand and selects a tradeoff between energy conservation and application quality, where application quality is represented in terms of the multimedia data presented to users. They have experimented with reducing the fidelity of video data, namely, frame size and resolution for multimedia applications, when the battery energy goes below a certain level. This approach can extend battery life by 30% while meeting quality goals.

Tan, Raghunathan and Jha [149] have proposed a set of *architectural transformations* to be applied to software applications designed with multi-process software style driven by an operating system. The resulting software design has been shown to be more energy efficient. The objectives of transformation techniques can be summarized as follows:

- *Reduce the number of processes:* Two periodic events occurring at the same can be handled in one process, rather than two, thereby saving energy, because of fewer context switches. In addition, two processes, that are executed sequentially because one passes data to the other, can be merged. This saves energy by executing fewer context switches and inter-process communication (IPC).
- *Reduce the number of IPC:* Opportunities should be exploited to merge code within a process so that miltiple IPC writes (or sends) are replaced with a single write (or send). This technique takes advantage of IPC calls that can accept multiple messages, for example, the *msg_snd()* system call in Linux. Fewer IPCs means less execution of code and less energy consumption.
- *Substitution of IPC:* A message passing-based IPC can be replaced with a shared memory operation with semaphore protection. Here the intuition is that message passing costs more energy than accessing shared memory.
- *Reduction of frequency of data transfer from device driver:* A device driver can buffer data for a while before sending it to a process. This technique reduces the number of times a process is activated, thereby saving energy.

They implemented the above techniques in the context of an embedded system featuring the StrongARM processor and the embedded Linux OS, and reported energy reduction up to 66.1%.

Heath et al. [187] have studied the potential benefits of application supported disk management for reducing energy consumption. They have proposed application transformations that increase disk idle times and inform the OS about the length of each upcoming period of idleness. The key ideas in application transformations are as follows:

- The code is modified to cluster disk read operations so that the processor could process a large amount of data in between two clusters of disk accesses.
- Care is exercised to not group all disk read operations into a single cluster, because this would increase buffer requirements.
- They propose the creation of a system call for an application to determine how much buffer is available to the application.
- The size of a read cluster is determined based on the size of the buffer space available to the application.

They have modeled the length of disk idle time and showed that there are significant benefits to performing those transformations for interactive applications. It may be noted that the transformations do not lead to any benefit for noninteractive applications. They have proposed a compiler framework that can transform applications automatically. With extensive experiments they have shown that the proposed transformations can achieve 70% disk energy savings with about 3% degradation in performance.

Chung et al. [188] have studied the impact of two well-known code transformation techniques on energy saving:

- **Loop unrolling:** This technique unfolds a loop by a certain number of times, called *unrolling factor*, and its aim is to reduce the number of processor cycles by reducing the loop overheads. The cost of loop unrolling is the increase in I-cache misses.
- **Loop blocking:** This breaks large arrays into several pieces (tiles smaller than cache size) and reuses each piece. The cost of loop blocking is the increase in the depth of loop nesting.

Therefore, the source code transformation approach offers some opportunities for energy saving while incurring some additional energy costs. Their experimental results show that the transformation approach finds the optimal energy saving transformations in 95% of the cases, and that the penalty when the non-optimal transformation is selected is within 5%.

XIII. Integrated Power Management

Power consumption has been modeled both at component level (e.g. [161], [163], [169]) and high level (e.g. [162], [160], [170], [173]). Several solutions for achieving energy efficiency have been proposed at various *computational* levels, namely, cache and external memory access optimization, dynamic voltage scaling (DVS), dynamic power management of disks and network interfaces, efficient compilers, and application/middleware based adaptations [176], [175]. Power optimization techniques developed at each computation level have remained largely independent of the other abstraction levels, thereby not exploiting the opportunities for further improvements achievable through cross-level integration. Lorch and Smith [171] have advocated the need for software techniques for taking advantage of low-power hardware. Specifically, developers of solutions to computer energy reduction should keep in mind:

- a hardware feature is rarely a complete solution since software modification is generally needed to make best use of it;
- energy consumption can be reduced not only by reducing the power consumption of components, but also by introducing low-power, low-functionality modes for those components;
- evaluation of a power management strategy should take into account not only how much energy it saves, but also whether the trade-off it makes between energy savings and performance is desirable for end-users; and
- seemingly independent energy management strategies can be made to interact.

Mohapatra et al. [177], [31] have proposed a model, as illustrated in Fig. 33, for integrating hardware based architectural optimization techniques with high level operating system and middleware approaches for video streaming applications. Multimedia applications havily utilize the biggest power consumers in modern handheld devices: CPU, WNIC, and display. Therefore, hardware and software techniques are integrated to induce power savings for those resources. They use quality and video encoding parameters of the video stream to optimize internal cache configurations, CPU registers, and external memory accesses. Knowledge of the underlying architectural details is used by the compiler to generate code that complements the optimizations at the low-level arhcitecture. Hardware design-level infromation (e.g. cache configuration) and user-level information (video quality perception) are utilized to optimize middleware and OS components for improved energy saving via effective video transcoding, power-aware admission control, and efficient transmission over the wireless link. Power consumption of the WNIC is reduced by putting it to sleep during inactive periods. The middleware controls network traffic for optimal power management of the WNIC. Tests have been performed to study video quality and power trade-offs for handheld devices. Their architectural optimizations saved 47.5 – 57.5% energy for the CPU and memory. The middleware technique enabled 60 – 78% savings in the power consumption of the WNIC.

Poellabauer and Schwan [158] have developed an energy saving approach that integrates a packet transmission policy to increase traffic burstiness, CPU task scheduling, dynamic frequency scaling (DFS), and scheduling packet transmission. Their approach works as follows:

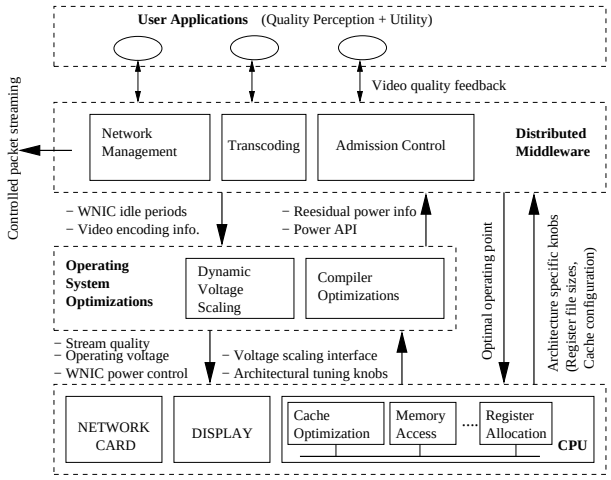


Fig. 33. Conjunctive Low-level and High-level Adaptations for Optimizing Power and Performance of Streaming Video to Mobile Handheld Computers [177].

- A QoS packet scheduler transmits packets according to their scheduling attributes: *ready times* t_r (earliest allowable transmission time) and *deadlines* t_d (latest allowable transmission time). Packets with no deadlines are transmitted *after* real-time packets.
- Packet transmissions are delayed without violating their t_d constraints to increase the likelihood of *bursty* transmissions. If no schedulable packets are in the packet queue, the WNIC is switched to a low-power *doze* mode.
- DFS is coordinated with data transmission such that when the packet scheduler's queue is empty, the lowest possible CPU clock frequency is used to delay the generation of new packets. As soon as the first packet is generated, the system switches to the highest possible clock frequency, in order to increase burstiness and increase data transmission.
- The real-time CPU task scheduler is adjusted such that tasks that have a low probability of packet generation are preferred while the packet queue is empty. When the packet queue is non-empty, the task scheduler schedules those tasks that have a high probability of packet generation, in order to increase data burstiness.

In summary, the above appraoch increases the time a WNIC is kept in doze mode and minimizes the number of switches between doze and active modes.

XIV. Energy Estimation Models

Modern wireless handheld systems are high in communication capabilities, namely, multiple wireless interfaces and bandwidth, and low in computing, storage, and battery energy. Such systems would not have all applications resident in their storage systems. Rather, applications would have to be downloaded from the network upon request [163]. For higher availability of handheld systems, the energy constrained devices should be able to estimate the energy cost of an application that has to be executed. The capability to estimate the energy cost of an application can be utilized to make subsequent decisions about its activities based on user input and sustainable battery life. If the remaining battery energy is low, the system can run the application at reduced throughput by reducing its voltage [164], or reduced accuracy [165]. Here, throughput is measured in the number of instructions executed per second. The energy-accuracy-throughput tradeoffs necessitate robust energy models for software based on target processors, operating clock frequency, and voltage. In this section, we discuss a number of energy estimation models with a variety of assumptions.

A. Low-level energy modeling: The key concept proposed by Sinha and Chandrakasan [163] is that the application that is downloaded has an associated energy model which the handheld system can use to configure itself based on user specification and energy availability. In order to reduce power consumption in low power hardware, supply voltages are constantly being lowered. However, to be able to maintain higher performance, the threshold voltage is also scaled down proportionately to provide sufficient current drive and reduce the propagation delay. For lower threshold voltage, the subthreshold leakage current becomes increasingly dominant. In reference [163] a simple energy model is presented for software based on frequency and supply voltage as parameters. The model incorporates explicit characterization of leakage energy as opposed to switching energy. A technique has been given to isolate the two components – leakage energy and switching energy – at the transistor level. More work needs to be done to carry the transistor-level model up to application layer. Kim et al. [169] have presented a way of fast estimation of software energy consumption in off-the-shelf processor using IPI (Inter-Prefetch Interval) energy model, where IPI is the time interval between two consecutive instruction prefetches. The IPI energy model is based on instruction-based energy models [161]. The reason they use prefetch information is to report not only total but also instantaneous energy consumption due to executing an application software. The IPI energy model is effective when information about target processor's behavior is not available. By reporting energy profile, the method offers an opportunity to find energy hotspots so that the model can be used in energy optimization. The concept of IPI energy model is useful to estimate energy cost of embedded processors.

B. High-level energy modeling: Tan et al. [160] have studied two high-level approaches to energy modeling: (i) complexity-based macro-modeling and (ii) profiling-based macro-modeling. The complexity-based approach is proposed for functions whose average-case computational complexities can be easily expressed in terms of some parameters (e.g. size of input data). For example, consider the insertion sort algorithm, which has an average-case complexity of $\Theta(n^2)$. For this function, the energy macro-model is as follows:

$$E = c_1 + c_2 s + c_3 s^2 \tag{20}$$

where s is the size of the input array. Regression analysis is used to obtain the unknown coefficients c_1, c_2, and c_3. For

three different input sizes s_1, s_2, and s_3, we have:

$$\begin{pmatrix} E_1 \\ E_2 \\ \dots \\ E_n \end{pmatrix} = \begin{pmatrix} 1 & s_1 & s_1^2 \\ 1 & s_2 & s_2^2 \\ \dots & \dots & \dots \\ 1 & s_n & s_n^2 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} \quad (21)$$

Equation (21) can be written in the form of $\mathbf{E} = \mathbf{P} \times \mathbf{C}$, where $\mathbf{C} = (c_1 \quad c_2 \quad c_3)^T$. $\mathbf{C}$ is obtained using regression analysis: $\mathbf{C} = [\mathbf{P}^T\mathbf{P}]^{-1}\mathbf{P}^T\mathbf{E}$. The macro-model is evaluated using the following relative error matric:

$$\epsilon = \sum_{i=1}^{n} \frac{((\hat{E}_i - E_i)/E_i)^2}{n} \quad (22)$$

where E_i's are the energy data values and $\hat{E}_i$'s are the values given by the macro-model.

On the other hand, the profiling-based macro-modeling technique is applied if the average-case complexity of a function is not known. A *basic block* is a maximal block of instructions such that when control enters the block at its first instruction, all its subsequent instructions are executed [161]. By measuring the energy cost of each basic block and counting the number of times each block is executed, one can estimate the total energy cost of a function as follows:

$$\hat{E} = c_1 b_1 + c_2 b_2 + \dots \quad (23)$$

where b_j's are the basic block execution counts and c_j's are the regression coefficients to be determined. However, the fitting error can be large. In order to improve the accuracy of estimation, the concept of *correlation profiling* has been introduced. Consecutive sequence of events are captured by the concept of correlation. Two kinds of correlation profiling, namely, *basic-block correlation* and *Ball-Larus path correlation* have been introduced in [160]. The cofficient constants in the basic-block correlation technique are derived from a set of linear equations similar to Eq. (21).

A "power data bank" approach to software energy estimation was proposed by Qu et al. [162]. They have exploited the fact that many embedded software programs are constructed using significant amount of pre-defined library packages and some glue code. The power data bank for a processor contains energy consumption and execution time values for basic instructions and library functions. A drawback of the power data bank approach is that different instances of a library function may have different energy consumption.

Mahmud et al. [178] have proposed a model for predicting energy consumption in the wireless network access portion of a handheld device equipped with multiple access technologies. Future handheld devices will be equipped with multiple wireless access technologies, namely, wireless personal area networks (e.g. Bluetooth, ZigBee, and Ultra Wide Band), wireless LAN (e.g. WiFi and HiperLAN), wireless metropolitan area network (WiMAX), cellular (e.g. 2G, 3G, and 4G), and broadcasting and multicasting systems (e.g. satellite). In IP based multi-service user terminals (MUT) with *always-on* capability, there is a need for an efficient addressing of the energy consumption issue. To be ennergy-efficient, a MUT should be equipped with a mechanism to choose the right interface for various types of communication requirements, downloading a file of certain size, for example. Based on the idea of minizing the energy cost of downloading a file, the authors give an algorithm to choose the appropriate interface while initiating a communication session and during handover. The algorithm makes a number of assumptions: (i) the energy parameters of all wireless interfaces are known; (ii) the device knows the available bandwidth for all the interfaces.

C. System-level energy modeling: Palit, Singh and Naik [170] present a model to estimate the energy cost of applications running on a portable wireless device. To develop the cost model, they partition a wireless device into two components, namely, computation (i.e. processing) and communication, as shown in Fig. 34 and Fig. 35, respectively. Each component is modeled by a state-transition diagram. Two attributes are associated with each state: an average power cost and a state residence time. The cost of each state of the state-transition diagrams is validated by actual measurements. For a constant voltage supply, the average power cost of a state is denoted by the average current drawn by the component. The state residence times are estimated from the behavior of applications. The cost model has been validated by performing actual measurement of energy cost. The estimation error is in the range of 5-10%, which is found out by performing actual measurement.

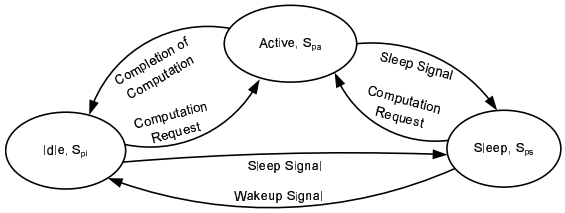


Fig. 34. Energy Cost Model of the Computation (i.e. Processing) Module of a Handheld System [170]. The "Idle" state in the Processing module is denoted by S_{pi}, and so on.

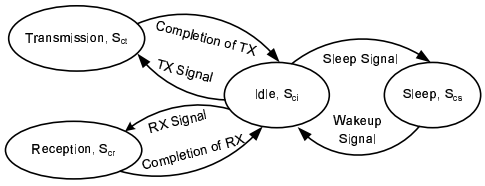


Fig. 35. Energy Cost Model of the Communication Part of a Handheld System [170] (TX: Transmission; RX: Reception). The "Idle" state in the Communication module is denoted by S_{ci}, and so on.

Assume that a task A runs for T seconds on the processing and communication modules. If the voltage supply v is constant for the execution interval T and the average current drawn during T is $I_{p,c}$, then the energy cost of A is given by the following equation:

$$E(A) = v \times I_{p,c} \times T. \quad (24)$$

Assuming that I_p and I_c are the average current drawn by the processing module and the communication module, respectively, we have:

$$I_{p,c} = I_p + I_c. \tag{25}$$

If the processor utilization is α, where $\alpha = \frac{\Delta tpa}{T}$ such that Δ_{tpa} is the total time spent by the processing module in state S_{pa} (i.e. the "Active" state), then we have:

$$I_p = \alpha I_{pa} + (1 - \alpha) I_{pi}. \tag{26}$$

The quantities I_{pa} and I_{pi} represent the average current drawn by the processing module in its "Active" and "Idle" states, respectively. Next, in order to get an expression for I_c, Palit, Singh, and Naik [170] apply the following technique. Let N_{tmax} and N_{rmax} stand for the maximum number of data packets that can be transmitted and received during the interval T, respectively. While running the application A, let N_t and N_r be the actual number of packets transmitted and received, respectively. Then, they define three quantities $\beta_t = N_t/N_{tmax}$, $\beta_r = N_r/N_{rmax}$, and $\beta_i = (1 - \beta_t - \beta_r)$. Now, an expression for I_c is as follows:

$$I_c = \beta_i I_{ci} + \beta_t I_{ct} + \beta_r I_{cr} \tag{27}$$

where $\beta_i I_{ci}$, and $\beta_t I_{ct}$, and $\beta_r I_{cr}$ stand for the average current drawn by the communication module it its "Idle," "Transmit," and "Receive" states, respectively. The model has been validated by perfoming measuremnets of the energy cost of three web based applications involving computation, downloading, and uploading. The estimated and the measured energy costs were within 5–10% of each other. No energy cost was overlooked in the measurement process because measurement was performed at the power supply level.

Min, Cha, and Ha [173] have introduced a system-level integrated power management (S-IPM) mechanism to reduce the total energy consumption of handheld devices. S-IPM uses the interactions between the CPU and the LCD as well as the WNIC to control the CPU voltage and frequency and the WNIC power mode. The method consists of a workload classification step and a power mode setting step. It classifies an application using information gathered through the WNIC. In the second step, S-IPM sets the LCD frequency and the WNIC power modes to correspond to the type of application and selects an appropriate voltage and frequency for CPU. The method reduces the system energy consumption by 12-23% on the average, compared to either no energy saving policy or conventional method of simply combining dynamic voltage scaling and dynamic power management.

Mahapatra, et al. [174] have presented a system level, dynamic game theoretic approach for choosing power optimization strategies for different components that draw energy from the same battery. They model the components, namely, CPU, WNIC, and LCD, as *players* in a non-cooperative game, and determined how each component should consume energy. They have evaluated two techniques: (i) components employ strategies that aim to maximize the overall utility of the system called *social optimum*; and (ii) each component uses a best-response strategy for maximizing its own utility. They have reported that strategies that achieved a socially optimal energy usage provided the maximum energy savings and with similar utility values.

Creus and Niska [155] have presented a *policy-based* approach for system-level power management on mobile devices. Hardware makers typically provide power saving features in their components. At the system-level, there is a need for binding those features to be able to take advantage of component-level energy efficiency. Policies provide flexibility in managing power at the system-level. Some examples of the concept of flexibility in power management are: (i) trade-off between energy efficiency and performance; (ii) allowing applications to communicate their intentions to the system through a set of application program interfaces (APIs); and (iii) collaboration among applications. The concept of a *Policy Manager* software design pattern by means of a *class diagram* in the Unified Modeling Language (UML) has been shown in Fig. 36. The ideas in the Policy Manager diagram (Fig. 36) are as follows:

- A policy is the translation of operational conditions to a concrete mode of operation of resource management and a concrete setting of the resource environment.
- The Policy Manager sets the type and amount of active resources based on the operational conditions of the system.
- The Policy Manager makes decisions based on input such as which applications are running and their operational mode, battery state of charge, and desired quality of service.

Figure 37 shows the logical architecture, in UML package notation, of a software entity for system-level power management. The concept of *policy* is at the core of the architecture. The Power Policy package represents an entity that makes decisions about the resources in the system. The Power Policy Manager takes into account all policies and makes final decisions for the complete system. The Resource Manager handles accounting and access control for the resources, and takes policies into account in handling the access to the resources. The Application Manager serves as a registry and entry point for all applications. Energy-aware applications send requests to and receive notifications from the framework concerning their state and resource usage. The Context Monitor is responsible for gathering all context-related information in the system and making it available to all parties in the framework. The framework described in Figs. 36 and 37 are useful in designing application software that can take advantage of operating system support for energy saving.

D. Characterization of Applications: Simulation (e.g. [168]), modeling (e.g. [161], [163], [160], [170]), and measurement (e.g. [170]) are three basic ways to evaluate the eneregy cost of software systems. Modeling can be performed at any level of abstraction, starting from the lowest,

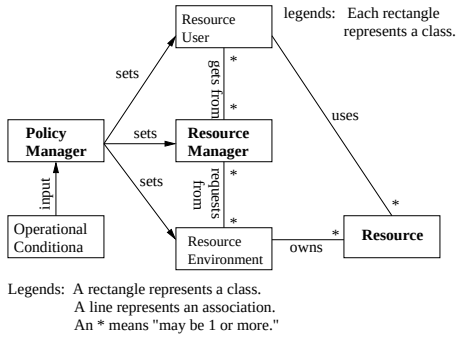


Fig. 36. Policy-driven Resource Manager Pattern [155].

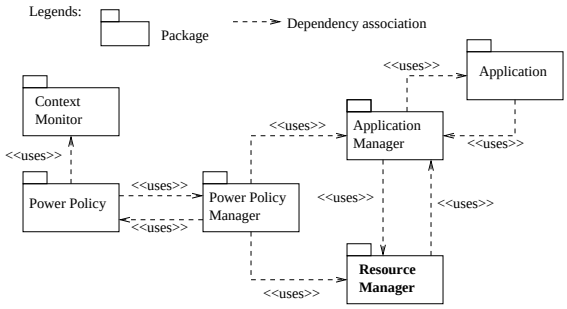


Fig. 37. Logical Architecture of a Power Management Framework [155].

instruction level [161] all the way up to the application level [170]. Detailed power simulation is time consuming and often inaccurate, whereas detailed power measurement is faster and objective. Fine-grained measurement generates a huge number of data in which locating important features is difficult, whereas coarse-grained measurement leaves out important details [166]. Hu, Jimenez and Kremer [166], [167] have presented an infrastructure to characterize *power behavior* of programs based on the idea of execution *phases*. They instrument the program at the assembly code level to enable precise power measurement of representative intervals in specific phases. Two key concepts in their work are counting of traversals of *control-flow edges*, instead of basic block vectors, and incorporating event counters into their phase classification. They have shown that the representative intervals accurately model the fine-grained time-dependent behavior of the program. Their results have been validated through actual measurement of benchmark programs on an Intel Pentium 4 system. They have shown that using *edge vectors* reduces the error of estimating total program enengy by 35% over using *basic block vectors*.

Other methods have also used application classifications to drive component-level policies [159]. The approach presented by Weissel et al. [159] focuses on reduction of energy consumption of the network interface of a handheld device as follows: (i) data traffic parameters gathered from send/receive statistics at the link layer-level are mapped to a predefined set of application profiles; and (ii) for an identified profile, an application-specific power management setting is triggered, A power management setting is defined by a *beacon interval*, which indicates the sleep period of the IEEE 802.11 wireless interface of a handheld device. Application profiles are characterized by the following parameters regarding network traffic: (i) average size of packets received; (i) average size of packets sent; (iii) ratio of average length of inactive to length of active periods; (iv) ratio of average size of packets received to size of pckets sent; (v) ratio of traffic volume received to traffic volume sent; (vi) standard deviation of the average size of packets receivced; and (vii) standard deviation of the average length of inactive periods. In a nutshell, the work of Weissel et al. [159] presents a power saving policy which dynamically adapts to the detected application profile. Their characterization algorithm was evaluated by running several typical applications for mobile devices: Mozilla web browser, Secure Shell (SSH) session, Network File System (NFS) operation, File Transfer Protocol (FTP) download, Netradio (low bandwidth real audio stream), MP3 audio stream, and high bandwidth real video stream.

The S-IPM method presented by Min, Cha and Ha [173] uses application classification in a system-wide context. By monitoring kernel behavior, S-IPM gathers information on badwidth utilization, packet size and port numbers from network traffic. Applications are classified based on their bandwidth utilization and burst level. A suggested appropriate listening interval (ALI) for the WNIC to guarantee an allowable quality of service is as follows:

$$ALI = c \times \frac{ABP \times PS \times RI \times ET}{ADR}, \qquad (28)$$

where c is a constant, ABP is the number of available buffered packets, PS is the packet size, and ET is the gap between the time stamps for sending and receipt and corresponding request and response packets. RI is related to user activity and determined as: $RI = TSS - TSS_{pre}$, where TSS is the current sending time stamp and TSS_{pre} the previous sending time stamp.

XV. Impact of DPM Strategies

The preceeding sections have shown that dynamic power management in handheld devices require additional computing elements in different locations across the network. The computing elements may run on the handheld devices, access points, servers, proxy machines, or a combination theirof. Those are implemented with special hardware and/or software, and the software components can be located over a broad spectrum, namely, from the kernel of an OS to application-level. In addition, the DPM code can span multiple protocol layers [192]. From a designer's viewpoint, two power requirements (PR) must be satisfied as follows:

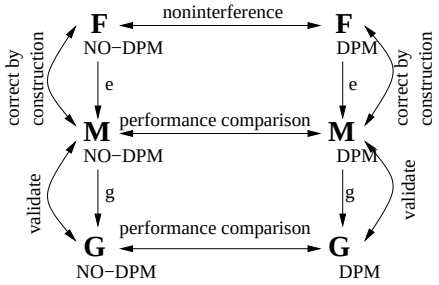
- *Noninterference (PR1):* Noninterference means that addition of the DPM component does not alter the functionality of the system. In other words, a system with a DPM component is functionally equivalent to one without the DPM.
- *Energy saving (PR2):* It is important to show that a DPM component indeed saves energy in handheld devices under a variety of operating conditions, such

as application types, device characteristics, and channel conditions. In other words, a DPM should be able to add value to handheld devices in general.

Requirement PR1 tells us that DPM does not become an obstacle for the device in delivering services. On the otherhand, PR2 states that a DPM strategy indeed enables a handheld device to reduce its energy cost. Acquaviva et al. [79] have proposed a incremental methodology, as illustrated in Fig. 38, to verify that a system supporting DPM satisfies both PR1 and PR2. In Fig. 38, **F**, **M**, and **G** stand for the functional, Markovian, and general models of a system under consideration. Each type of model appears in two forms: without DPM support (i.e. NO-DPM) and with DPM support. The lefthand column lists all three models (i.e. **F**, **M**, and **G**) with NO-DPM, whereas the righhand column lists all the three models with DPM. The models were described in the *Aemilia* [199] architectural description language using the *TwoTowers* tool [200]. *Aemilia* was chosen for its expressiveness to describe both functionality and performance aspects of systems. The accompanying *TwoTowers* tool is useful in comparing two models written in *Aemilia*.

As illustrated in Fig. 38, in the beginning, two functional models (**F**) – one with NO-DPM and one with DPM – of the system under study are described in Aemilia. The two functional models are compared using noninterference analysis techniques [198], [197] that use the concept of observational equivalence. Next, the two functional models are enriched with the specification of timing of each system activity through exponentially distributed random variables. As a result, in the second phase, Markovian models (**M**) are obtained from the corresponding **F** models. That is, one **M** model with NO-DPM and one **M** model with DPM are obtained in the second phase. An **M** model need not be validated against its corresponding **F** model because it is derived by attaching rate information to the state transitions of the **M** model. The two **M** models are solved anlytically and compared using a variety of performance metrics: energy consumption, throughput, and radio channel utilization, to name a few. Finally, the exponential delays in the two **M** models are replaced with generally distributed delays in order to observe the general energy saving capability of the DPM strategy. The **G** models are validated against their **M** models to observe if the two demonstrate the same level of average performance. If the validations suceed, the two **G** models are simulated to evaluate the energy saving potential of the DPM strategy under different realistic scenarios. Usefulness of the incremental methodology has been studied by applying it on two applications, nanmely, *remote procedure call* and *streaming video service*.

Kim et al. [202] have used the *Real-time Maude* language to express application requirements, dynamicity of environment, and desired correctness constraints to analyze power policies. *Real-time Maude* is a language and tool for the formal specification, simulation, and analysis of real-time systems. A system state is specified as a multi-set of objects and messages, and system behavior is represented by two



Legends: e = add exponential timing; g = replace with general timing
F = Functional model; M = Markovian model; G = General model

Fig. 38. An Incremental Methodology to Evaluate the Impact of DPM Strategies [79].

kinds of *rewrite* rules: *unconditional rules* and *conditional rules*. Essentially, rules describe state tansitions that can be instantaneous or timed. Rewriting and timed search are used to generate execution traces to analyze the system by model checking with temporal logic. They have shown how to use *Real-time Maude* to specify and analyze four power policies:

- *Always-on:* This policy ignores energy saving.
- *Greedy:* The power manager shuts down the device whenever its idle period goes beyond a threshold.
- *Cluster:* The power manager tries ro aggregate idle periods to maximize energy saving.
- *DVS:* The operating voltage is dynamically scaled down.

Qiu, Wu and Pedram [201] have used Generalized Stochastic Petri Net (GSPN) to model a DPM for systems with complex characteristics: concurrency, synchronization, mutual exclusion, and conflict. A GSPN model is automatically transformed into an equivalent continuous-time Markov model. The optimal power management policy for components are computed by solving a linear programming problem based on the Markov model.

XVI. Summary

We are witnessing tremendous growth in wireless communication technologies in terms of coverage, data rate, and communication range in both the licensed and free spectrums. Advances in processor, memory, storage, display, and user interface technologies are enabling system designers to develop ever smaller and more powerful handsets with multi-radio interfaces supporting celluar, Wi-Fi, and Bluetooth connections. Modern handheld devices in the family of smart phones and PDAs, for example BlackBerry and iPhone, support both voice and data communications for making phones calls and browsing the Internet. On the other hand, application specific handheld devices, for example, iPod for music, iPad for electronic book reading and phone calls, and Kindle for electronic book reading, are on the rise. Almost universal Internet accessibility and world-wide mobility make the pocket-size devices so popular for general communication, entertainment, and business applications. Small size, which makes handheld devices so popular, also imposes resource constraints on the devices in their processing power,

memory and storage capacity, and battery life time. For handheld devices, the energy resource plays a vital role in determining their dependability, because once a device's battery is completely depleted it becomes unavailable for further use. A number of software based strategies have been developed over the past 15 years to make handheld devices energy efficient. For a comprehensive understanding of the energy saving approaches available in the reseach literature, we organized about 200 reserach articles into 14 sections.

In Section III, we explained the key idea behind smart batteries, the architectural details of a smart battery, and how to make applications aware of the remaining charge quantity of the battery. Knowledge about the SoC level of the battery can be used in designing energy-aware applications. It is important for programmers to understand battery characteristics to be able to schedule tasks so as to reduce peak energy demand.

In Section IV, we reviewed the important reserach results in energy efficient displays and GUI design. For displays, we explained the concepts of *backlight control*, *frame buffer compression*, *dynamic control of color depth* and *refresh cycle*, and *dark window optimization*. For efficient user interactions, we explained the roles of hardware, operating system, and application programs. We explained the concept of a *model human processor* and three fundamental processes, namely, *perception capacity*, *cognitive capacity*, and *motor speed*. Three groups of specific energy reduction techniques, namely, *power reduction*, *performance enhancement*, and *facilitators*, in GUI design were discussed. Finally, we explained the KLEM method for predicting user interaction time and energy cost.

In Section V, we explained the concept of a $\mu Sleep$ state of an OS and the concept of *wake-on-wireless*. The $\mu Sleep$ state of an OS is different from its *Idle* state, and it is an indication to the procssor to go into a sleep state thereby saving more energy. The concept of *wake-on-wireless* aims to eliminate the power consumed when a Wi-Fi enabled device is idle. The *wake-on-wireless* concept requires a second wireless interface, one or more proxy, and a device server. The main drawback of this approach is the need for a large number of proxy machines to be deployed.

In Section VI, we discussed the power saving techniques built into the IEEE 802.11 family of standards for Wi-Fi networks, a few novel power saving techniques for IEEE 802.11 based devices with dynamic ATIM window management and longer sleep time, MAC-level download scheduling by APs, and power management in WiMAX subscriber stations. All the approaches to energy saving focused on putting a user device's WNIC into its sleep state for the maximum amount of time.

A number of proxy assisted energy saving techniques were discussed in Section VII. Those proxies were *data transforming proxy* with power-friendly data transformation and transmission, *HTTP-level power aware web proxy*, *power aware streaming proxy*, *caching proxy* for streaming media delivery, and *streaming audio proxy*, We explained the network architectures and protocols for those proxies, and discussed their energy saving potential.

Proxyless, server-based approaches to energy saving in handheld devices were discussed in Section VIII. Servers can perform *traffic adaptation* and *traffic shaping* to enable handheld devices to save energy by utilizing information about applications: input buffer size, expected rate at which the user device empties data from the buffer, WLAN card on/off transition time, and WLAN card status. We discussed the details of a software architecture for server controlled power management. The section included two scheduling policies, namely, *Closed-Loop Dynamic Power Management* and *Open-Loop Dynamic Power Management*, to exploit the information available at the application level to save energy.

Transport protocols are widely used by handheld devices to download files from servers. Therefore researchers have explored several ways to save energy in handheld devices by evaluating the computational energy cost of transport protocols and controlling the energy cost of WNIC by utilizing TCP states. Those results and the concept of a power-saving transport protocol were presented in Section IX.

Energy saving techniques that can be applied at the application level have been studied in Section X. Those techniques are *data compression*, *download scheduling*, and *computation offloading*. Handheld devices have relatively less amount of stable storage compared to, say, laptops. The concept of *virtual* memory is key to designing applications with high storage requirements. With high latency of stable storage devices in handheld systems and with incresingly faster network interfaces, system designers may be tempted to implement a network-based virtual memory system. The efficacy of implementing a network-based virtual memory system, from the standpoint of energy efficiency, has been discussed in Section XI.

In Section XII, we summarized a number of low-level programming techniques in order to save the energy execution cost of applications. Those techniques focused on *reducing memory accesses*, *energy cost driven code generation*, *instruction reordering for low power*, and *instruction packing* and *dual memory loads*. In addition, we explained a software environment for energy-aware programming where three key concepts were energy related *API library*, *energy data recording*, and *energy prediction* for individual tasks.

In Section XIII, we explained a software architecture that enables designers of handheld systems to integrate individual energy saving techniques and strategies. Because, in general, an energy saving technique for one subsystem cannot be applied independent of other components, it is important to integrate individual energy saving techniques at the levels of packet transmission for burstiness, CPU task scheduling, dynamic frequency scaling, coding and compilation, and displays.

Energy cost prediction is key to making handheld devices adaptive to the remaining amount of energy. In Section XIV, we reviewed a number of energy cost modeling approaches,

namely, *low-level energy modeling*, *high-level energy modeling*, *system-level energy modeling*, and *characterization of applications*. In Section XV we identified the issues to be considered after incorporaring a DPM component into a system. Specifically, there is a need to verify that the added component does not interfere with the original functionality of the system and that the DPM component indeed enables the system to save energy. We explained how high-level system description languages, namely, *Aemilia*, *Real-time Maude*, and *Generalized Stochastic Petri Nets*, can be used to model and analyze systems with a DPM component.

XVII. Future Directions

In spite of the tremendous progress made to achieve energy efficiency in wireless handheld devices, much more can be done as we come to the realization that battery energy defines the *availability* of the device. In this section, we identify a few opportunities that augment the current strategies for energy efficiency and point to new research directions.

- *Battery State Information:* The concept of wireless channel state information (CSI) [207] allows us to understand how the wireless link is behaving from the perspective of packet loss. It is used in developing down-link packet scheduling algorithms to maximize link throughput by identifying the user facing the best channel quality at a given time. Similarly, the concept of battery state information (BSI) can be developed at the OS level to maximize the battery life-time of the device. BSI will include the present charge capacity of the battery and the rates at which major subsystems are draining energy. The OS can use the information to schedule tasks for energy efficiency, and also provide the BSI information to energy-aware applications. Low-cost current sensing hardware [208] is needed to gather BSI information from batteries. Battery models (e.g., see [7] and [14]) can be applied to construct macro-models of BSI from the microdata about current.
- *Coordinated Strategies:* The existing energy saving strategies work independently, with an implicit assumption that those are orthogonal, meaning that their energy saving effects are additive. It will be interesting to validate that those strategies and techniques can coexist to produce additive benefits. More interesting will be any identification of their their synergistic effects in reducing energy costs.
- *Efficient task scheduling:* The existing strategies and techniques for energy efficiency largely ignore the long-term life management of batteries. There is a need for new processor scheduling algorithms to: (i) maximize the lengths of its idle periods; and (ii) reduce the current peaks. The first objective will let the process sleep longer, whereas the second will slow down the decay of battery capacity. For example, coordination among the major subsystems of a handheld device can reduce current peaks.
- *Exploit the resources available in the personal workspace:* It is not uncommon for users to carry multiple portable wireless systems, namely, a smartphone and a laptop. New strategies and techniques can be developed to enhance the battery life of a smartphone at the expense of the laptop [209]. This paradigm of resource sharing is different than the proxy- and server-based approaches.
- *System-wide policy and mechanism to accumulate small data packets:* We studied the traffic pattern of smartphones and observed that a good portion of the packets are of small size and the generated traffic is bursty in nature. Motivated by these two observations, we have proposed a Low Energy Data-packet Aggregation Scheme (LEDAS) [210]. It accumulates a number of upper layer packets into a burst at the WiFi MAC level, based on formation time, size, and number of packets. With this scheme, larger bursts lead to longer inactivity periods during which the communication module can be kept in doze mode. In addition, fewer MAC frames lead to less overheads and contentions in the wireless medium. However, the data packets incur delays due to the accumulation process. We have evaluated the efficacy of the technique by simulations and showed the energy-delay trade-offs. The idea can be extended to MAC protocols based on slotted time and short packets.

REFERENCES

[1] R. Rao, S. Vrudhula and D. N. Rakhmotov, "Battery Modeling for Energy-Aware System Design," *IEEE Computer*, Dec. 2003, pp. 77-86.

[2] R. Casas and O. Casas, "Battery Sensing for Energy-Aware System Design," *IEEE Computer*, Nov. 2005, pp. 48-54.

[3] P. S. Marshall, "Brains and Brawns – The Power of Smart Batteries," *Sensors Magazine*, April 2002.

[4] S. Piller, M. Perrin, and A. Jossne, "Methods for State-of-Charge Determination and Their Applications," *Journal of Power Sources*, June 2001, pp. 113-120.

[5] D. Rakhmatov and S. Vrudhula, "Energy Management for Battery-Powered Embedded Systems," *ACM Trans. on Embedded Computing Systems*, Vol. 2(3), August 2003, pp. 277-324.

[6] D. Rakhmatov, S. Vrudhula and D. A. Wallach, "A Model for Battery Lifetime Analysis for Organizing Applications on a Pocket Computer," *IEEE Trans. on VLSI Systems*, 2003, Vol. 11(6), pp. 1019-1030.

[7] V. Rao, G. Singhal, A. Kumar and N. Navet, "Battery Model for Embedded Systems," *Proc. 18th Int. Conf. on VLSI Design*, 2005, pp. 105-110.

[8] N. Zheng, Z. Wu, M. Lin and Q. Wang, "An Iterative Computation Method for Interpreting and Extending an Analytical Battery Model," *Proc. of the 20th IEEE Int. Conf. on VLSI Design*, 2007, pp. 601-608.

[9] R. Rao, S. Vrudhula and N. Chang, "Battery Optimization vs Energy Optimization: Which to Choose and When?," *IEEE/ACM ICCAD*, 2005, pp. 438-444.

[10] T. L. Martin et al., "A Case Study of a System-Level Approach to Power-Aware Computing," *ACM Trans. on Embedded Computing Systems*, Vol. 2(3), August 2003, pp. 255-276.

[11] M. R. Jongerden and B. R. Haverkort, "Which Battery Model to Use?," *IET Software*, Vol. 3(6), 2009, pp. 445-457.

[12] M. R. Jongerden, B. R. Haverkort, H. Bohnenkamp and J. P. Katoen, "Maximizing System Lifetime by Battery Scheduling," *Proc. 39th Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks* (DSN'09), 2009, pp. 63-72.

[13] P. Rong and M. Pedram, "An Analytical Model for Predicting the Remaining Battery Capacity of Lithium-Ion Batteries," *IEEE Trans. on VLSI Systems*, Vol. 14(5), 2006, pp. 441-451.

[14] D. Rakhmatov, "Battery Voltage Modeling for Portable Systems," *ACM Trans. on Design Automation of Electronic Systems*, Vol. 14(2), 2009.

[15] R. A. Jackey, G. L. Plett and M. J. Klein, "Parameterization of a Battery Model Using Numerical Optimization Methods," *SAE International*, 2009-01-1381.

[16] L. Benini et al., "Discrete-Time Battery Models for System-Level Low-Power Design," *IEEE Trans. on VLSI Systems*, Vol. 9(5), Oct. 2001, pp. 630-640.

[17] W. M. Saslow, "How Batteries Discharge: A Simple Model," *American Journal of Physics*, 76(3), March 2008, pp. 218-223.

[18] T. F. Fuller, M. Doyle and J. Newman, "Simulation and Optimization of the Dual Lithium Ion Insertion Cell," *Journal of Electrochemical Society*, 1994, Vol. 141(1), pp. 1-10.

[19] S. C. Hageman, "Simple PSpice Models Let You Simulate Common Battery Types," *Electronics Design News*, Vol. 38, 1993, pp. 117-129.

[20] C. Chiasserini and R. Rao, "Pulsed Battery Discharge in Communication Devices," *Proc. 5th Int. Conf. Mobile Computing and Networking*, 1999, pp. 88-95.

[21] C. Chiasserini and R. Rao, "A Model for Battery Pulsed Discharge with Recovery Effect," *Proc. Wireless Comm. and Networking Conference*, 1999, pp. 636-639.

[22] C. Chiasserini and R. Rao, "Improving Battery Performance by Using Traffic Shaping Techniques," *IEEE Journal on Selected Areas in Communications*, 2001, Vol. 19(7), pp. 1385-1394.

[23] C. Chiasserini and R. Rao, "Energy Efficient Battery Management," *IEEE Journal on Selected Areas in Communications*, 2001, Vol. 19(7), pp. 1235-1245.

[24] J. Manwell and J. McGowan, "Lead Acid Battery Storage Model for Hybrid Energy Systems," *Solar Energy*, 1993, Vol. 50, pp. 399-405.

[25] A. Sears and B. Schneiderman, "Split Menus: Effectively Using Selection Frequency to Organize Menus," *ACM Trans. on Computer-Human Interaction*, Vol. 1, No. 1, March 1994, pp. 27-51.

[26] P. M. Fitts, "The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement," *Journal of Experimental Psychology*, Vol. 47, No. 6, June 1954, pp. 381-391.

[27] B. B. Bederson, "Fisheye Menus," *Proc. of Symp. User Interface Software and Technology*, Nov. 2000, pp. 217-225.

[28] M. T. Raghunath and C, Narayanaswami, "User Interfaces for Applications on a Wrist Watch," *Personal Ubiquitous Computers*, Vol. 6, No. 1, 2002, pp. 17-30.

[29] W.-C. Cheng, Y. Hou and M. Pedram, "Power Minimization in a Backlit TFT-LCD Display by Concurrent Brightness and Contrast Scaling," *IEEE Trans. on Consumer Electronics*, Vol. 50, No. 1, Feb. 2004, pp. 25-32.

[30] H. Shim, N. Chang and M. Pedram, "A Backlit Power Management Framework for Battery Operated Multimedia Systems," *IEEE Design & Test of Computers*, Vol. 21, No. 5, May 2004, pp. 388-396.

[31] S. Mohapatra and N. Venkatasubramanian, "Proactive Energy-Aware Video Streaming to Mobile Handheld Devices," *Proc. of Int. Conference on Multimedia Communication and Networking*, 2002.

[32] H. Shim, N. Chang and M. Pedram, "A Compressed Frame Buffer to Reduce Display Power Consumption in Mobile Systems," *IEEE/ACM Asia South Pacific Design Automation Conference*, 2004, pp. 819-824.

[33] S. Iyer, R. Mayo and P. Ranganathan, "Energy Adaptive Display System Designs for Future Mobile Environments," *The 1st International Conference on Mobile Systems, Applications and Services*, 2003.

[34] H. Shim, Y. Cho and N. Chang, "Power Saving in Hand-held Multimedia Systems Using MPEG-21 Digital Item Adaptation," *IEEE Workshop on Embedded Systems for Real-time Multimedia*, 2004, pp. 13-18.

[35] H. Shim, Y. Cho and N. Chang, "Frame Buffer Compression Using a Limited-Size Code Book for Low-Power Display Systems," *IEEE Workshop on Embedded Systems for Real-time Multimedia*, 2005, pp. 7-12.

[36] J. Gong and P. Tarasewich, "Guidelines for Handheld Mobile Devices Interface Design," *Proc. of the Decision Sciences Institute (DSI) 2004 Annual Meeting*.

[37] J.-W. Hyun, I. Kumazawa and M. Sato, "A New Measurement Methodology of Multi-Finger Tracking for Handheld Device Control Using Mixed Reality," *SICE Annual Conference*, 2007, pp. 2315-2320.

[38] J. Hyun, I. Kumazawa and M. Sato, "The Design Method for User Interface of Handheld Devices Using SPIDAR-Hand System," *IEICE Tech. Rep.*, Vol. 108, No. 374, pp. 1-6, Jan. 2009.

[39] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," *Proc. of 15th Annual ACM Symposium on Principles of Distributed Computing*, 1996, pp. 1-7.

[40] S. Brewster, "Overcoming the Lack of Screen Spaces on Mobile Computers," *Personal and Ubiquitous Computing*, Vol. 6, 2002, pp. 188-205.

[41] N. Chang, I. Choi and H. Shim, "DLS: Dynamic Backlight Luminance Scaling of Liquid Crystal Display," *IEEE Trans. on VLSI Systems*, Vol. 3, No. 8, August 2004, pp. 837-846.

[42] S. Pasricha et al., "Dynamic Backlight Adaptation for Low-Power Handheld Devices," *IEEE Design & Test of Computers*, Sept.-Oct. 2004, pp. 398-405.

[43] A. Schmidt, "Implicit Human Computer Interaction Through Context," *Personal Technologies*, 4(2 & 3), 2000, pp. 191-199.

[44] I. Choi, H. Shim and N. Chang, "Low-Power Color TFT LCD Display for Hand-Held Embedded Systems," *Proc. of ACM/IEEE Int. Symposium on Low-Power Electronics Design*, 2002, pp. 112-117.

[45] F. Gatti, A. Acquaviva, L. Benini and B. Ricco, "Low Power Control Techniques for TFT LCD Displays," *Int. Conf. on Compilers, Architecture and Synthesis for Embedded Systems*, 2002, pp. 218-224.

[46] L. Zhong and N. K. Jha, "Graphical User Interface Energy Characterization for Handheld Computers," *CASES'03*, pp. 232-242.

[47] L. Zhong and N. K. Jha, "Energy Efficiency of Handheld Computer Interfaces: Limits, Characterization and Practice," *ACM Int. Conference on Mobile Systems, Applications and Services*, 2005, pp. 247-260.

[48] K. S. Vallerio, L. Zhong and N. K. Jha, "Energy-Efficient Graphical User Interface Design", *IEEE Trans. on Mobile Computing*, Vol. 5, No. 7, July 2006, pp. 846-858.

[49] B. A. Myers and M. B. Rosson, "Survey on User Interface Programming," *Proc. of ACM Conference on Human Factors in Computing Systems*, 1992, pp. 195-202.

[50] J. Johnson, "GUI Bloopers: Don'ts and Do's for Software Developers and Web Designers," Morgan Kaufmann, San Francisco, CA, 2000.

[51] S. K. Card, T. P. Moran and A. Newell, "The Psychology of Human-Computer Interaction," Hillsdale, N. J.: Lawrence Erlbaum Associates, 1983.

[52] R. P. Carver, "Reading Rate: A Review of Research and Theory," Academic Press Inc., San Diego, California, 1990.

[53] W. E. Hick, "On the Rate of Gain of Information," *Quarterly Journal of Experimental Psychology*, No. 4, 1952, pp. 11-36.

[54] R. Hyman, "Stimulus Information as a Determinant of Reaction Time," *Journal of Experimental Psychology*, No. 46, 1953, pp. 188-196.

[55] L. Luo, "Designing Energy and User Efficient Interactions with Mobile Systems," *PhD Dissertation, CMU-ISR-08-102*, Carnegie Mellon University, 2008.

[56] L. Luo and D. P. Siewiorek, "KLEM: A Method for Predicting User Interaction Time and System Energy Consumption During Application Design," *11th International Symposium on Wearable Computers*, Boston, 2007.

[57] L. Luo and B. E. John, "Predicting Task Execution Time on Handheld Devices Using the Keystroke-Level Model," *Proc. CHI 2005*, Portland.

[58] D. Siewiorek, "New Frontiers of Application Design," *Communications of the ACM*, 45(12), Dec. 2002.

[59] S. K. Card, T. P. Moran and A. Newell, "The Keystroke-Level Model for User Performance Time with Interactive Systems," *Communications of the ACM*, 23(7), 1980, pp. 396-410.

[60] "http://www.cogtool.org/download.html" CogTool website.

[61] "http://act-r.psy.cmu.edu/" The ACT-R website.

[62] B. John, K. Prevas, D. Salvucci and K. Koedinger, "Predictive Human Performance Modeling Made Easy," *Proc. CHI*, 2004.

[63] S. Iyer, L. Luo, R. Mayo and P. Ranganathan, "Energy-Adaptive Display System Designs for Future Mobile Environments," *First Int. Conference on Mobile Systems, Applications and Services (MobiSys)*, 2003.

[64] G. Anastasi, M. Conti and M. D. Francesco, "Energy Conservation in Wireless Sensor Networks: A Survey," *Ad Hoc Networks*, Vol. 7, No. 3, May 2009, pp. 537-568.

[65] P. Shenoy and P. Radkov, "Proxy-Assisted Power Friendly Streaming to Mobile Devices," *Proc. of the 2003 Multimedia Computing and Networking Conference*.

[66] M. C. Rosu, C. M. Olsen, C. Narayanaswami and L. Luo, "PAWP: A Power Aware Web Proxy for Wireless LAN Clients," *IEEE Workshop on Mobile Computing Systems and Applications*, 2004, pp. 206-215.

[67] M. C. Rosu, C. M. Olsen, L. Luo and C. Narayanaswami, "The Power-Aware Streaming Proxy Architecture," *First Int. Workshop on Broadband Wireless Multimedia (BroadWim)*, 2004.

[68] S. Chen, H. Wang, B. Sen, S. Wee and X. Zhang, "Segment-based Proxy Caching for Internet Streaming Media Delivery," *Proc. of IEEE Multimedia*, 2005, pp. 59-67.

[69] A. Acquavivaa, T. Simunic, V. Deolalikar and S. Roy, "Server Controlled Power Management for Wireless Portable Devices," *HP-Labs, HPL-2003-82*.

[70] G. Anastasi, et al., "An Energy-Efficient Protocol for Multimedia Streaming in a Mobile Environment," *Journal of Pervasive Computing and Communications*, Dec. 2005.

[71] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, "RTP: Real Time Protocol," *IETF RFC 3550, http://tools.ietf.org/html/rfc3550, July 2003*.

[72] H. Schulzrinne, A. Rao and R. Lanphier, "RTSP: Real Time Streaming Protocol," *IETF RFC 2326, http://www.ietf.org/rfc/rfc2326.txt, April 1998*.

[73] M. Handley, S. Floyd, J. Padhye and J. Widmer, "TFRC: TCP-Friendly Rate Control Protocol," *IETF RFC 3448, http:// www.ietf.org/ rfc/ rfc3448.txt*, Jan. 2003.

[74] M. Handley, S. Floyd, J. Padhye and J. Widmer, "RTCP: Real Time Control Protocol," *IETF RFC 3605, http://www.ietf.org/rfc/rfc3605.txt*, Oct. 2003.

[75] G. Anastasi, M. Conti, I. Giannetti and A. Passarella, "Design and Evaluation of a BitTorrent Proxy for Energy Efficiency," *IEEE Symposium on Computers and Communications*, July 2009.

[76] Z. Miao and A. Ortega, "Scalable Proxy Caching of Video Under Storage Constraints," *IEEE Journal of Selected Areas in Communications*, Vol. 20, Sept. 2002, pp. 1315-1327.

[77] Y. Chae, K. Guo, M. Buddhikot, S. Suri and E. Zegura, "Silo, Rainbow and Caching Token: Schemes for Scalable Fault Tolerant Stream Caching," *IEEE Journal of Selected Areas in Communications*, Vol. 20, Sept. 2002.

[78] A. Acquaviva, T. Simunic, V. Deolalikar and S. Roy, "Remote Power Control of Wireless Network Interfaces," *Journal of Embedded Computing*, Vol 3, 2005, pp. 381-389.

[79] A. Acquaviva, A. Aldini, M. Bernardo, et al., "Assessing the Impact of Dynamic Power Management on the Functionality and the Performance of Battery-Powered Appliances," *IEEE Int. Conf. on Dependable Systems and Networks*, 2004.

[80] A. Acquaviva, E. Lattanzi and A. Bogliolo, "Design and Simulation of Power-Aware Scheduling Strategies of Streaming Data in Wireless LANs," *ACM MSWiM*, 2004.

[81] P. T. Breuer, A. M. Lopez and A. G. Ares, "The Network Block Device," *Linux Journal*, 73, May 2000.

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

[83] S. A. Akella, R. K. Balan and N. Bansal, "Protocols for Low-Power," *Tech. Report*, Carnegie-Mellon University, 2001.

[84] D. Bertozzi, L. Benini and B. Ricco, "Power Aware Network Interface Management for Streaming Multimedia," *Proc. IEEE Wireless Communications and Networking Conference*, March 2002.

[85] T. Simunic, L. Benini, P. W. Glynn and G. D. Micheli, "Dynamic Power Management for Portable Systems," *Proc. ACM Int. Conf. on Mobile Computing and Networking*, 2000, pp. 11-19.

[86] A. Seddik-Ghaleb, Y. Ghamti-Doudane and S. Senouci, "TCP Computational Energy Cost Within Wireless Mobile Ad Hoc Network," *The 7th ACS/IEEE International Conference on Computer Systems and Applications*, Morocco, 2009.

[87] A. Seddik-Ghaleb, Y. Ghamri-Doudane and S. Senouci, "A Performance Study of TCP Variants in Terms of Energy Consumption and Average Goodput within a Static Ad Hoc Environment," *International Conference on Wireless Communication and Mobile Computing*, Vancouver, 2006, pp. 503-508.

[88] A. Seddik-Ghaleb, Y. Ghamri-Doudane and S. Senouci, "Emulating End-to-End Losses and Delays for Ad Hoc Networks," *IEEE ICC*, Glasgow, 2007.

[89] Network Simulator–2 (NS-2). Available at www.isi.edu/nsnam/ns/.

[90] S. Agrawal and S. Singh, "An Experimental Study of TCP's Energy Consumption over a Wireless Link," *4th European Personal Mobile Communications Conference*, 2001.

[91] V. Jacobson, "4BSD Header Prediction," *ACM Computer Communication Review*, Vol. 20, No. 1, April 1990, pp. 13-15.

[92] S. Bansal, R. Gupta, R. Shorey, A. Misra, I. Ali and A. Razdan, "Energy Efficiency and Throughput for TCP Traffic in Multi-Hop Wireless Networks," *IEEE INFOCOM*, 2002.

[93] S. Bansal, R. Shorey, R. Gupta and A. Misra, "Energy Efficiency and Capacity of TCP Traffic in Multi-Hop Wireless Networks," *Wireless Networks*, Vol. 12(1), Feb. 2006.

[94] H. Singh and S. Singh, "Energy Consumption of TCP Reno, New-Reno, and SACK in Multi-hop Wireless Networks," *ACM SIGMETRICS*, June 2002.

[95] B. Wang and S. Singh, "Computational Energy Cost of TCP," *IEEE INFOCOM*, March 2004.

[96] D. Bertozzi, A. Raghunathan, L. Benini and S. Ravi, "Transport Protocol Optimization for Energy Efficient Wireless Embedded Systems," *Proc. of Design Automation and Test Conference*, 2003, pp. 706-711.

[97] M. Conti, L. Donatiello, F. Mazzoni and A. Zobbi, "Exploiting an Indirect TCP Model for Power Saving: Remarks and Experimental Results," *Proc. MoMuC*, 1998, pp. 213-222.

[98] R. Kravets and P. Krishnan, "Power Management Techniques for Mobile Communication," *Proc. MOBICOM*, 1998, pp. 157-168.

[99] Y. Lee, H. Lee, H, Shin, "Adaptive Spatial Resolution Control Scheme for Mobile Video Applications," *IEEE Int. Symposium on Signal Processing and Information Technology*, 2007.

[100] S. Ma, W. Gao and Y. Lu, "Rate-distortion Analysis for H.264/AVC Video Coding and its Application to Rate Control," *IEEE Trans. on Circuit and Systems Video Technology*, Vol. 15, Dec. 2005, pp. 1533-1544.

[101] J. Ribas-Corbera and S. Lei, "Rate Control in DCT Video Coding for Low-delay Communications," *IEEE Trans. on Circuit and Systems Video Technology*, Vol. 9, Feb. 1999, pp. 172-185.

[102] Y. Takishima, M. Wada, H. Murakami, "An Analysis of Optimal Frame Rate in Low Bit Rate Video Coding," *IEICE Trans. on Communications*, Vol. E76-B, Nov. 1993, pp. 1389-1397.

[103] J. Kim, Y.-G. Kim, H. Song, T.-Y. Kuo, Y. J. Chung and C.-C. J. Kuo, "TCP-friendly Internet Video Streaming Employing Variable Frame Rate Encoding and Implementation," *IEEE Trans. on Circuit and Systems Video Technology*, Vol. 10, Oct. 2000, pp. 1164-1177.

[104] H. Song and C.-C. J. Kuo, "Rate Control for Low-bit-rate Video via Variable-encoding Frame Rates," *IEEE Trans. on Circuit and Systems Video Technology*, Vol. 11, April 2001, pp. 512-520.

[105] Y. Yuan, D. Feng and Y. Zhong, "Fast Adaptive Variable Frame-rate Coding," *IEEE VTC*, Fall'2004, pp. 2734-2738.

[106] L. S. Brakmo, D. A. Wallach, and M. A. Viredazand, "μSleep: A Technique for Reducing Energy Consumption in Handheld Devices," HP Lab, Palo Alto, HPL-2004-11, 2004.

[107] IEEE Standard for Wireless LAN – Medium Access Control and Physical Layer Specification, Nov. 1997.

[108] E. Shih, P. Bahl, and M. J. Sinclair, "Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices," *ACM MOBICOM* 2002.

[109] E.-S. Jung and N. H. Vaidya, "An Energy Efficient MAC Protocol for Wireless LANs," *Proc. of IEEE INFOCOM*, 2002, Vol. 3, pp. 1756–1764.

[110] H. Woesner, J.-P. Ebert, M. Schlager and A. Wolisz, "Power-saving Mechanisms in Emerging Standards for Wireless LANs: The MAC-level Perspective," *IEEE Personal Communications*, Vol. 5, 1998, pp. 40-48.

[111] S. Singh, M. Woo and C. S. Raghavendra, "Power-aware Routing in Mobile Ad Hoc Networks," *ACM MOBICOM*, 1998, pp. 181-190.

[112] J. C. Cano and P. Manzoni, "Evaluating the Energy Consumption Reduction in a MANET by Dynamically Switching Off Network Interfaces," *6th IEEE Symp. on Computers and Communications*, July 2001, pp. 186-191.

[113] W. Ye, J. Heidermann and D. Estrin, "An Energy-efficient MAC Protocol for Wireless Sensor Networks," *IEEE INFOCOM*, 2002, pp. 1567-1576.

[114] J.-M. Choi, Y.-B. Ko and J.-H. Kim, "Enhanced Power Saving Scheme for IEEE 802.11 DCF Based Wireless Networks," *IFIP International Conference on Personal Wireless Communications*, Sept. 2003, pp. 835-840.

[115] E.-S. Jung and N. H. Vaidya, "Improving IEEE 802.11 Power Saving Mechanism," *Wireless Networks* (2008) Vol. 14(3), pp. 375-391.

[116] R. Kravets and P. Krishnan, "Application-driven Power Management for Mobile Communication," *Wireless Networks*, 6(4), 2000, pp. 263-277.

[117] A. Bakre and B. R. Badrinath, "Implementation and Performance Evaluation of Indirect TCP," *IEEE Trans. on Computers*, 46(3), March 1997, pp. 260-278.

[118] M. Stemm and R. H. Katz, "Measuring and Reducing Energy Consumption of Network Interfaces in Handheld Devices," *IEICE Trans. Fund. Electornics Comm. Com. Sc.*, 80(8), 1997, pp. 1125-1131.

[119] H. Yan, R. Krishnan, S. A. Watterson and D. K. Lowenthal, "Client-centered Energy Savings for Concurrent HTTP Connections," *ACM NOSSDAV*, 2004.

[120] R. Krashinsky and H. Balakrishnan, "Minimizing Energy for Wireless Web Access with Bounded Slowdown," *Wireless Networks*, 11(1-2), Jan. 2005, pp. 135-148.

[121] D. Qiao and K. G. Shin, "Smart Power-saving Mode for IEEE 802.11 Wireless LANs," *IEEE INFOCOM*, 2005.

[122] S. Nath, Z. Anderson and S. Seshan, "Choosing Beacon Periods to Improve Response Times for Wireless HTTP Clients," *ACM MobiWac*, 2004.

[123] M. Anand, E. Nightangle and J. Flinn, "Self-tuning Wireless Network Power Management," *Wireless Networks*, 11(4), July 2005, pp. 451-469.

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

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

[126] G. Anastasi, M. Conti, E. Gregori and A. Passarella, "802.11 Power-saving Mode for Mobile Computing in Wi-Fi Hotspots: Limitations, Enhancements and Open Issues," *Wireless Networks* (2008) 14:745–768.

[127] IEEE 802.16e, "IEEE Draft Standard for Local and Metropolitan Area Networks – Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems. Amendment for Physical and MAC Layers for Combined Fixed and Mobile Operation in Licensed bands," Feb. 2007.

[128] Y. Xiao, "Energy Saving Mechanism in the IEEE 802.16e Wireless MAN," *IEEE Comm. Letters*, Vol. 9, No. 7, 2005, pp. 595-597.

[129] Y. Zhang and M. Fujise, "Energy Management in the IEEE 802.16e MAC," *IEEE Comm. Letters*, Vol. 10, No. 4, 2006, pp. 311-313.

[130] J. Jang, K. Han and S. Choi, "Adaptive Power Saving Strategies for IEEE 802.16e Mobile Broadband Wireless Access," *Proc. APCC*, 2006.

[131] M.-G. Kim, M. Kang and J.-Y. Choi, "Performance Evaluation of the Sleep Mode Operation in the IEEE 802.16e MAC," *Proc. ICACT*, 2007

[132] Y.-H. Han, S.-G. Min and D. Jeong, "Performance Comparison of Sleep Mode Operations in IEEE 802.16e Terminals," *Proc. ICCS*, 2007, pp. 441-448.

[133] D. T. T. Nga, M. Kim and M. Kang, "A Novel Energy Saving Algorithm with Frame Response Delay Constraint in IEEE 802.16e," *IEICE Transactions on Communications*, Vol. E91-B(4), 2008, pp. 1190-1193.

[134] G. Anastasi et al., "Performance Evaluation of Power Management for Best Effort Applications in IEEE 802.16 Networks," *Proc. of European Wireless*, 2008.

[135] J. Lee, C. Rosenberg and E. K. P. Chong, "Energy Efficient Scheduler Design in Wireless Networks," *Proc. of Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2004, pp. 237-246.

[136] J. Lee, C. Rosenberg and E. K. P. Chong, "Energy Efficient Schedulers in Wireless Networks: Design and Optimization," *Mobile Networks and Applications*, Vol. 11, 2006, pp. 377-389.

[137] P. J. M. Havinga and G. J. M. Smit, "QoS Scheduling for Energy-Efficient Wireless Communication," *Proc. ITCC*, Las Vegas, 2001.

[138] P. J. M. Havinga and G. J. M. Smit, "Energy-efficient Wireless Networking for Multimedia Applications," *Wireless Communications and Mobile Computing*, Vol. 1(2), 2001, pp. 165-184.

[139] J. A. Stine and G. de Veciana, "Energy Efficiency of Centrally Controlled Wireless Data Networks," *Wireless Networks*, Vol. 8, No. 6, 2002, pp. 681-700.

[140] J.-C. Chen, K. M. Sivalingam, P. Agrawal and S. Kishore, "A Comparison of MAC Protocols for Wireless Local Networks Based on Battery Power Consumption," *Proc. INFOCOM*, 1998, pp. 150-157.

[141] S. Singh and C. Raghavendra, "Power Efficient MAC Protocol for Multihop Radio Networks," *Proc. of PIMRC Conference*, 1998, pp. 153-157.

[142] K. M. Sivalingam, J.-C. Chen, P. Agrawal, "Design and Analysis of Low-power Access Protocols for Wireless and Mobile ATM Networks," *ACM/Baltzer Wireless Networks*, Vol. 6(1), Feb. 2000, pp. 73-87.

[143] K. M. Sivalingam, M. Srivastava and P. Agrawal, "Low Power Link and Access Protocols for Wireless Multimedia Networks," *Proc. IEEE VTC*, May 1997, pp. 1331-1335.

[144] I. Chlamtac, C. Petrioli, J. Redi, "Energy-conserving Access Protocols for Identification Networks," *IEEE/ACM Transactions on Networking*, Vol. 7, No. 1, Sept. 1999, pp. 51-59.

[145] R. Xu, Z. Li, C. Wang and P. Ni, "Impact of Data Compression on Energy Consumption of Wireless-Networked Handheld Devices," *IEEE International Conference on Distributed Systems*, 2003.

[146] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Trans. on Information Theory*, 23(3), 1977, pp. 337-343.

[147] M. Burrows and D. J. Wheeler, "A Block-sorting Lossless Data Compression Algorithm," *Tech. Report 124*, Digital Systems Research Center, Palo Alto, CA, May 1994.

[148] T. A. Welch, "A Technique for High-performance Data Compression," *IEEE Computer*, 17(6), 1984.

[149] T. K. Tan, A. Raghunathan, and N. Jha, "Software Architectural Transformations: A New Approach to Low Energy Embedded Software," *Proc. of the Conference on Design Automation and Test in Europe* (DATE'03), 2003.

[150] A. Rudenko, P. Reiher, G. J. Popek and G. H. Kuenning, "Saving Portable Computer Battery Power Through Remote Process Execution," *ACM Mobile Computer Communication Review*, 2(1), 1998, pp. 19-26.

[151] Z. Li, C. Wang and R. Xu, "Computation Offloading to Save Energy on Handheld Devices: A Partition Scheme," *Proc. Int. Conf. on Compilers, Architectures and Synthesis for Embedded Systems*, 2001, pp. 238-246.

[152] C. Wang and Z. Li, "Parametric Analysis for Adaptive Computation Offloading," *ACM SIGPLAN 2004 Conf. on Prog. Language Design and Implementation*, pp. 119-130.
[153] C. Wang and Z. Li, "A Computation Offloading Scheme on Handheld Devices," *Journal of Parallel and Distributed Computing*, Vol. 64, No. 6, 2004, pp. 740-746.
[154] J. Bang-Jensen and G. Gutin, *Diagraphs: Theory, Algorithms and Applications*, Springer, London, 2001.
[155] G. B. i Creus and P. Niska, "System-level Power Management for Mobile Devices", *Proc. of the 7th Int. Conference on Computer and Information Technology*, 2007, pp. 799-804.
[156] A. Acquaviva, E. Lattanzi and A. Bogliolo, "Power-Aware Network Swapping for Wireless Palmtop PCs," *IEEE Trans. Mob. Computing*, Vol. 5, 2006, pp. 571-582.
[157] R. Cornea, A. Nicolau and N. Dutt, "Software Annotations for Power Optimization on Mobile Devices," *Proc. DATE*, 2006, pp. 684-689.
[158] C. Poellabauer and K. Schwan, "Energy-Aware Traffic Shaping for Wireless Real-Time Applications," *Proc. RTAS*, 2004, pp. 48-55.
[159] A. Weissel, M. Faerber and F. Bellosa, "Application Characterization for Wireless Network Power Management," *Int. Conference on Architecture of Computing Systems*, 2004.
[160] T. K. Tan, A. Raghunathan, G. Lakshminarayana, and N. K. Jha, "High-level Software Energy Macro-modeling," *ACM DAC* 2001.
[161] V. Tiwari, S. Malik and A. Wolfe, "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization," *IEEE Trans. VLSI Systems*, Vol. 2(4), pp. 437-445, Dec. 1994.
[162] G. Qu, N. Kawabe, K. Usami, and M. Potkonjak, "Functional-level power estimation methodology for microprocessors," *In Proc. Design Automation Conf.*, 2000, pp. 810-813.
[163] A. Sinha and A. P. Chandrakasan, "Energy Aware Software," *13th International Conference on VLSI Design*, 2000.
[164] V. Gutnik and A. Chandrakasan, "An Embedded Power Supply for Low Power DSP," *IEEE Trans. on VLSI Systems*, Vol. 5, no. 4, Dec. 1997, pp. 425-435.
[165] S. H. Nawab et al., "Approximate Signal Processing," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, Vol. 15, no. 1/2, Jan. 1997, pp. 177-200.
[166] C. Hu, D. A. Jimenez and U. Kremer, "Efficient Program Power Behavior Characterization," *Proc. HiPEAC*, 2007, (LNCS 4367).
[167] C. Hu, D. A. Jimenez and U. Kremer, "An Evaluation Architecture for Power and Energy Optimisations," *Int. Journal of Embedded Systems*, Vol. 3, Nos. 1/2, 2007, pp. 31-42.
[168] D. Brooks, V, Tiwari and M. Martonosi, "Wattch: A Framework for Architectural Level Power Analysis and Optimizations," In Proc. *Int. Symposium on Computer Architecture*, 2000, pp. 83–94.
[169] J. Kim, K. Kang, H. Shim, W. Hwangbo and C.-M. Kyung, "Fast Estimation of Software Energy Consumption Using IPI (Inter-Prefetch Interval) Energy Model," *IFIP WG 10.5 International Conference on Very Large Scale Integration of System-on-Chip*, 2007, pp. 224-229.
[170] R. Palit, A. Singh and K. Naik, "Modeling the Energy Costs of Applications on Portable Devices," *the 11th ACM Int. Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, October 2008, Vancouver.
[171] J. R. Lorch and A. J. Smith, "Software Strategies for Protable Computer Energy Management," *IEEE Personal Communications*, June 1998, pp. 60-73.
[172] T. Newhall, S. Finney, K. Ganchev and M. Spiegel, "Nswap: A Network Swapping Module for Linux Clusters," *Proc. Euro-Par*, 2003.
[173] J.-H. Min, H. Cha and R. Ha, "System-level Integrated Power Management for Handheld Systems," *Microprocessors and Microsystems 33* (2009), pp. 201-210.
[174] S. Mohapatra and N. Venkatasubramanian, "A Game Theoretic Approach for Power Aware Middleware," *Middleware 2004, LNCS 3231*, pp. 417-438.
[175] S. Mohapatra and N. Venkatasubramanian, "PARM: Power-Aware Reconfigurable Middleware," *IEEE ICDCS*, 2003.
[176] P. Pillai and K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systesms," *Proc. of SOSP*, 2001.
[177] S. Mohapatra et al., "Integrated Power Management for Video Streaming to Mobile Handheld Devices," *11th ACM International Conference on Multimedia*, 2003, pp. 582-591.
[178] K. Mahmud, M. Inoue, H. Murakami, M. Hasegawa H. Morikawa, "Energy Consumption Measurement of Wireless Interfaces in Multi-Service User Terminals for Heterogeneous Wireless Networks," *IEICE Trans. on Communications*, Vol. E88-B, No. 3, 2005, pp. 1097-1110.
[179] T. Simunic, H. Vikalo, P. Glynn, and G. De Micheli, "Energy Efficient Design of Portable Wireless Systems," *Proc. of the 2000 Int. Symposium on Low Power Electronics and Design*.
[180] S. Gitzenis and N. Bambos, "Joint Task Migration and Power Management in Wireless Computing," *IEEE Trans. on Mobile Computing*, Vol. 8(9), September 2009, pp. 1189-1204.
[181] J. Flinn and M. Satyanarayanan, "Energy-aware Adaptation for Mobile Applications," *17th ACM Symposium on Operating Systems Principles*, 1999, pp. 48-63.
[182] U. Kremer, J. Hicks and J. M. Rehg, "A Compilation Framework for Power and Energy Management on Mobile Computers," *14th Intl. Workshop on Languages and Compilers for Parallel Comp.*, 2001.
[183] C. Xian, Y-H. Lu and Z. Li, "A Programming Environment with Runtime Energy Characterization for Energy-Aware Applications," *Proc. of ISLPED*, 2007, pp. 141-146.
[184] V. Tiwari, S. Malik, A. Wolfe, and M. T-C. Lee, "Instruction Level Power Analysis and Optimization of Software," *9th Intl. Conf. on VLSI Design*, 1996, pp. 326-328.
[185] T. C. Lee, V. Tiwari, S. Malik, and M. Fujita, "Power Analysis and Low Power Scheduling Techniques for Embedded DSP Software," *Intl. Symp. on System Synthesis*, Cannes, 1995.
[186] V. Tiwari, S. Malik, and A. Wolfe, "Compilation Techniques for Low Energy: An Overview," *IEEE Symp. on Low Power Electronics*, 1994.
[187] T. Heath, E. Pinheiro, J. Hom, U. Kremer and R. Bianchini, "Code Transformations for Energy-efficient Device Management," *IEEE Trans. on Computers*, 53(8), Aug. 2004, pp. 974-987.
[188] E. Y. Chung, L. Benini, and G. D. Micheli, "Source Code Transformation Based on Software Cost Analysis," *Proc. ISSS 2001*, Montreal, pp. 153-158.
[189] K. Naik and D. Wei, "Software Implementation Strategies for Power Conscious Systems," *ACM Mobile Networks and Architectures*, 6(3), pp. 291-305, 2001.
[190] V. Raghunathan, T. Pering, R. Want, A. Nguyen and P. Jensen, "Experience With a Low Power Wireless Mobile Computing Platform," *ACM/IEEE Int. Symposium on Low Power Electronics and Design*, 2004, pp. 363-368.
[191] J. Zhang, D. Wu and S. Ci, H. Wang and A. K. Katsaggelos, "Power-Aware Mobile Multimedia: a Survey," *J. of Communications*, Vol. 4(9), Oct. 2009, pp. 600-613.
[192] B. Bougard, S. Pollin, A. Dejonghe, F. Catthoor and W. Dehaene, "Cross-layer Power Management in Wireless Networks and Consequeneces on System-level Architecture," *Signal Processing (Elsevier)*, Vol. 86 (2006), pp. 1792-1803.
[193] E. Uysal-Biyikoglu, B. Prabhakar and A. E. Gamal, "Energy-Efficient Packet Transmission Over a Wireless Link," *IEEE/ACM Trans. on Networking*, Vol. 10(4), August 2002, pp. 487-499.
[194] S. Pollin et al., "Optimizing Transmission and Shutdown for Energy-Efficient Real-time Packet Scheduling in Clustered Ad Hoc Networks," *EURASIP J. on Wireless Comm. and Networking*, 2005:5, pp. 698-711.
[195] S. Padhye et al., "Modeling TCP Reno Performance, a Simple Model and Its Empirical Validation," *IEEE/ACM Trans. on Networking*, Vol. 8(2), April 2000, pp. 133-145.
[196] L. Benini, A. Bogliolo and G. de Micheli "A Survey of Design Techniques for System-Level Dynamic Power Management," *IEEE Trans. on VLSI Systems*, Vol. 8, 2000, pp. 299-316.
[197] R. Focardi and R. Gorrieri, "A Classification of Security Properties," *Journal of Computer Security*, Vol. 3, 1995, pp. 5-33.
[198] J. A. Goguen and J. Meseguer, "Security Policy and Security Models," *Symposium on Security and Privacy*, 1982, pp. 11-20.
[199] M. Bernardo, L. Donatiello and P. Ciancarini, "Stochastic Process Algebra: From an Algebraic Formalism to an Architectural Description Language," *Proc. Performance Evaluation of Complex Systems: Techniques and Tools*, LNCS 2459, 2002, pp. 236-260.
[200] M. Bernardo, "TwoTowers 3.0: Enhancing Usability," *Proc. MASCOTS 2003*, pp. 188-193.
[201] Q. Qiu, Q. Wu and M. Pedram, "Dynamic Power Management of Complex Systems Using Generalized Stochastic Petri Nets," *Proc. DAC*, 2000, pp. 352-356.
[202] M. Kim, N. Dutt and N. Venkatasubramanian, "Policy Construction and Validation for Energy Minimization in Cross Layered Systems: A Formal Method Approach," *Proc. of 12th Real-Time and Embedded Technology and Apps. Symp.*, 2006, pp. 25-28.
[203] J. Adams and G.-M. Muntean, "Adaptive-buffer Power Save Mechanism for Mobile Multimedia Streaming," *Proc. IEEE ICC*, 2007, pp. 4548-4553.

[204] S. Chandra and A. Vahdat, "Application-specific Network Management for Energy-Aware Streaming of Popular Multimedia Formats," *Proc. Usenix Annual Tech. Conference*, 2002.

[205] W. Yuan and K. Nahrstedt, "Energy-efficient CPU Scheduling for Multimedia Applications," *ACM Trans. on Computer Systems*, Vol. 24(3), 2006, pp. 292-331.

[206] D. Wu, S. Ci and H. Wang, "Cross-layer Optimization for Video Summary Transmission over Wireless Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 25(4), 2007, pp. 841-850.

[207] M. Mehrjoo, X. Shen and K. Naik, "A Channel-aware and Queue-aware Scheduling for IEEE 802.16 Wireless Metropolitan Area Networks," *Proc. of IEEE Wireless Communications and Networking Conference*, 2007.

[208] H. Vahedi, R. Muresan and K. Naik, "High Precision System-on-Chip Energy Management for Battery Lifetime Optimization," *Proc. 15th IEEE Int. Conf. on Electronics, Circuits, and Systems*, Athens, Greece, 2010.

[209] A. Singh, R. Palit and K. Naik, "An Architecture for Enhancing the Energy Efficiency and Capability of Handheld Devices," submitted to *WoWMoM* 2011.

[210] R. palit, K. Naik and A. Singh, "Impact of Packet Aggregation on Energy Consumption in Smartphones," submitted to *IWCMC*, 2011.