Optimal Hibernation Policies for Energy Efficient Stateful Operation in High-end Wireless Sensor Nodes

Carlo Brandolese, William Fornaciari, Luigi Rucco
Politecnico di Milano – DEIB
Piazza L. Da Vinci, 32 – 20133 Milano, Italy
{brandole,fornacia,rucco}@elet.polimi.it

***Abstract*—This paper proposes and studies an hibernation technique and optimal hibernation policies aimed at minimizing the power consumption, while allowing stateful processing and the adoption of more powerful nodes. To this purpose the paper models the energy trade-off for hibernating the system rather than putting it in a memory-retention sleep mode between two consecutive bursts of processing. Thanks to a simplified notion of system state, the paper formally determines the optimal conditions for deciding whether to hibernate or not the system during idle periods. Hibernation policies have been implemented as a module of the operating system and results demonstrate energy savings up to 50% compared to trivial hibernation approaches. Moreover, the hibernation policy proved to be robust and stable with respect to changes of the application parameters.**

I. Introduction

Today, after more than thirty years of research on distributed sensor networks, of which at least fifteen mainly focused on Wireless Sensor Networks, a broad diffusion of this technology outside academia and specialized niches cannot be appreciated yet. The causes of this low acceptance should be mainly sought in the well known limitations affecting this class of systems, from the harsh resource constrained hardware and software, to the need for long lifetime and minimal energy consumption, as well as the high complexity of the radio aspects, both in terms of energy and reliability. The high cost of specialization required by this technology discourages their adoption by potential investors. As a consequence, both private and public institutions are often resorting to universities and research centers asking for solutions which can mask this complexity and make WSNs more practical for commercial applications. In this direction moved the work we presented in [1], which proposed a joint development of hardware and software, especially at Operating System level, providing to the end user features similar to those of a general-purpose programmable system: a truly standard programming language, standard libraries, standard processes and threads, standard synchronization methods and programming paradigms, memory isolation and protection, application linking and loading. The operating model is inspired to the principle of leveraging data processing inside the network, pointed out since the very beginning of research on WSNs [2] and relying on the consideration—already true—that thousands or millions of operations per second can be done using the energy of sending a bit over a few tens of meters [3]. This principle helps in strongly reducing the energy consumption at network-wide level, considering the energy spared also by avoiding the cost of information forwarding from leaf and intermediate nodes to the base station. This entails the evolution of processing paradigms towards more complete applications, capable of locally analyzing data and triggering events and communications only when strictly needed. From a technological point of view, this requires full-fledged stateful processes and hardware platforms that can provide adequate support: the framework proposed in this work aims at this goal by introducing an intelligent hibernation mechanism and a formal model for optimal hibernation policy.

II. Related Work

Duty cycling has been widely explored as a power management technique in Wireless Sensor Networks. According to Anastasi et al. [4], duty-cycling can be classified in Topology Control and Power Management techniques, the last in turn divided in two main branches: sleep/wakeup protocols and low duty-cycle MAC protocols. In Topology Control techniques the duty cycle is distributed and consists of keeping awake the smallest possible subset of nodes needed to guarantee the connectivity, while bringing the others in sleep mode. Moreover, also the radio duty cycle is optimized by maintaining the transceiver active only when needed. Topology Control techniques are complementary to local duty cycle management and fall beyond the scope of the present work. We will focus, hence, on sleep/wakeup protocols. Many approaches have been proposed and to classify them we will follow the framework of Anastasi et al., which divides this class of models in three schemes, namely on-demand , asynchronous and scheduled rendezvous. Our approach falls in the last scheme, in particular we suppose that data are transmitted during the data processing period of the operating model.

III. Operating model

In the considered operating model we can logically distinguish two phases: *data processing* and *data sensing*. In particular, data processing consists of analyzing a given set of sampled data (and possibly transmit a result), once a given number of samples have been collected or when a certain threshold of the monitored parameter is reached. Operations of data sensing, on the other hand, entail sampling a certain sensor with an opportune frequency and push the measure into a queue that will be analyzed during the next processing phase. Reading a sensor and adding the retrieved measure to a queue are simple operations, such that maintaining the system in full active mode during this phase is indeed a waste of

978-1-4673-5828-6/13/$31.00

energy. So a possible solution is to hibernate the system once no more processes are performing data processing. During the hibernation period, some data sensing operations may be required with a fine-grained timing: in this case, the system may be resumed in a low power mode and with a lower clock frequency, just to perform sensor readings and put the measure in the associated queue. This operation is very fast (typically less than one millisecond) and does not require the boot of the entire operating system, neither to resume the process in charge of analyzing the entire set of data. Sensing operations are performed by a special module, called *smart sensing*, which, together with the hibernation manager, composes the power management infrastructure of the system.

IV. Hibernation scheme

In this work we will refer to *processes* as generic instances of a running program, regardless of their actual implementation, that can be either a complete process, a task, a thread, and so on. To avoid considering trivial operation of the system, we will concentrate on periodic processes or asynchronous ones, which wait for events that occur with a certain statistical distribution. We assume that the entire hibernation process is managed through two demons: an hibernation daemon and a resume daemon. The *Hibernation daemon* keeps track of the status of all the processes and once all are in a blocked state, takes the decision whether to either hibernate or entering sleep mode with memory retention, according to the hibernation policy. The hibernation daemon, moreover, saves the process descriptors in a fixed area of the non-volatile memory, and finally, when hibernating, swaps-out the memory images of the processes on the external memory. The *Resume daemon*, activated by an external real-time clock, swaps-in from the external non-volatile memory to the main internal memory the images of the processes that must be run and recreates their execution contexts in the operating systems. Note that the status of the operating system does not need to be saved, as it can be reconstructed from descriptors and images.

V. Model overview

Assuming that processes are either periodic or deterministic aperiodic, at a given time t_k it is possible to determine both the next wake-up time and which processes will be active at that time. We define the state of the system at time t_k as a boolean vector: $S_k = [s_{k,1}\ s_{k,1}\ \ldots\ s_{k,p}]$, where $s_{k,j} = 0$ if process q_j is currently swapped-out onto the external non-volatile memory, while $s_{k,j} = 1$ if the process is residing in RAM. The evolution of the system over time can thus be modeled as a sequence of pairs

$$(t_0, S_0),\ (t_1, S_1),\ \ldots,\ (t_k, S_k),\ \ldots \tag{1}$$

indicating the current absolute time and the state of all processes. For a periodic system the evolution is know for the entire future, while for a system with periodic and aperiodic processes it is know only until the latest known wake-up time of the aperiodic processes. It is possible to demonstrate [5] that the following global condition, referred to as $\mathcal{C}_n$, guarantees that hibernating is convenient in the time interval τ_k between the current time t_k and the next resume time t_{k+1}:

$$\mathcal{C}_n: \quad \tau_k > f(n) = \frac{||S_k||E_o + ||S_k \wedge S_{k+1}^{(n)}||E_i + E_{OS}}{P_S} \tag{2}$$

where E_i and E_o represent the energy consumption to swap a single byte in and out, respectively, while $P_S$is the power consumption of the system in sleep mode with memory retention. E_{OS} is the energy to shut-down and to boot the operating system. The *norm* of a state S_k is defined as $||S_k|| = S_k \cdot W$ (where the sizes' vector W is left implicit) and $S_{k+1}^{(n)}$ has been defined as the cumulate memory status for the next n states, hypothesizing that the system will never hibernate between S_{k+1} and S_{k+n}. If the condition $\mathcal{C}_n$ is not satisfied, then putting the system in sleep with memory retention would be the best choice. Studying the function $f(n)$ we observe that it has an absolute maximum f_M and this occurs as soon as $S_{k+1}^{(n)} \supseteq S_k$. The function also has a lower bound f_L, when $S_k \wedge S_{k+1}^{(n)} = \emptyset$. Considering these two values we can define two sufficient (but not necessary) conditions for determining when hibernation, rather than sleep with memory retention, is certainly convenient. In particular, when:

$$\mathcal{C}_M: \quad \tau_k > \frac{||S_k||(E_o + E_i) + E_{OS}}{P_S} \tag{3}$$

then the decision to hibernate is guaranteed to be optimal. We refer to this inequality as *global maximum condition*, or $\mathcal{C}_M$.

On the other hand, when:

$$\mathcal{C}_L: \quad \tau_k < \frac{||S_k||E_o + E_{OS}}{P_S} \tag{4}$$

the decision to sleep with memory retention is optimal. We will refer to this inequality as *global minimum condition*, or $\mathcal{C}_L$. In the general case, that is with a specific look-ahead of n states, the value of $f(n)$ will necessarily fall between its theoretical maximum and minimum. As anticipated, the specific value of n to be used for a particular combination of processes, energies, sizes and for each specific state cannot be determined in general, as it requires an exhaustive analysis of the system evolution, which, moreover, is only possible for periodic processes. We thus introduce the threshold f_α as:

$$f_\alpha = f_L + \alpha(f_M - f_L) \tag{5}$$

with $0 \leq \alpha \leq 1$. This leads to the new, general, condition:

$$\mathcal{C}_\alpha: \quad \tau_k > f_\alpha \tag{6}$$

where α can be determined experimentally in order to minimize the energy consumed by the system.

VI. Reference implementation

The hibernation mechanism has been implemented in MiosixOS, on the PoliNode prototype infrastructure [1]. This platform is based on a STM32F2 microcontroller, an external MR25H10 magnetoresistive RAM and an ultra low-power transceiver. The hibernation and resume daemons, described in Section IV, have been implemented as kernel threads running in background.

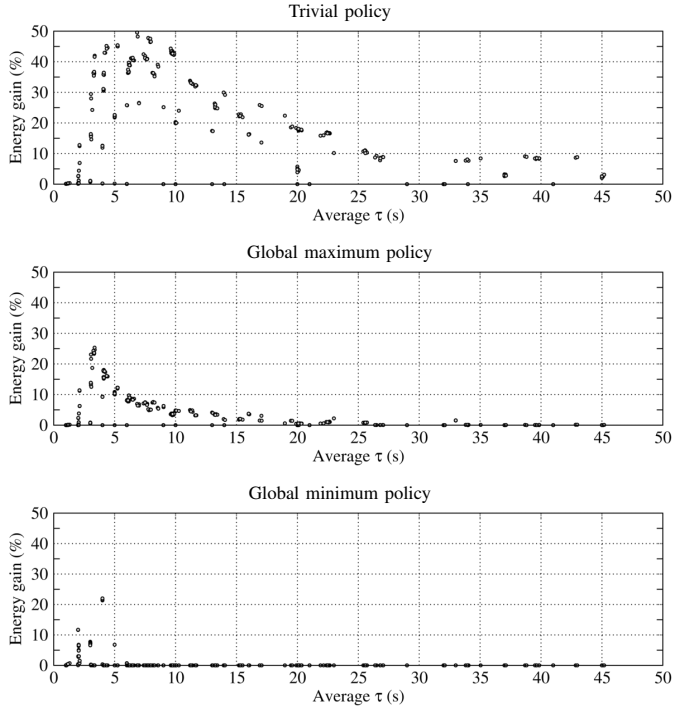

Fig. 1. Optimal versus sub-optimal policies energy gains

VII. Results

A highly optimized simulation flow has been implemented in C++ to meet the severe computational requirements of the stress tests and stability analysis on the simulated evolution of the real system. Most of the experiments reported in the following refer to the hardware/software characteristics described in Section VI, while others have been performed by perturbing the ratio E_{active}/P_{sleep}, the most significant parameter affecting hibernation choices. In the experiments we assumed a maximum available memory of 128 KB and a pool of 10 periodic processes, with memory image sizes randomly chosen between 1 KB and 32 KB. The RAM memory footprint of the specific operating system is about 32 KB, but it does not need to be swapped-out in the external MRAM since it performs a complete reboot each time the system is resumed. The first experiment assesses the energy gain obtained applying our policy with respect to three sub-optimal policies, namely: a *trivial* policy assuming to swap-in and out the entire memory, the *global maximum* and the *global minimum* policies defined by the conditions $\mathcal{C}_M$ and $\mathcal{C}_L$. The gains are reported in of Figure 1 against the average time interval τ between subsequent states. Each point corresponds to different combination of process periods and sizes and summarizes the average gain over 1,000 randomly generated tests. The trivial policy, often adopted for small sensor nodes, resulted by far the least efficient one. The gains with respect to the global minimum policy are smaller than those for the other policies, meaning that the global minimum policy well captures a greater number of cases if compared to the trivial and the global maximum policy. The parameter α strongly affects the gain. The robustness of the model in reference to this parameter has been assessed with a million tests, randomly changing the periods and sizes of the processes and by perturbing the ratio E_{active}/P_{sleep} of a factor 2, 4, 8, 12, 16 and 20. The policy has a good stability, converging towards the two attractors $\alpha = 0$, before a certain critical average period $\overline{p}_{min}$, and $\alpha = 1$, after a critical average period $\overline{p}_{max}$. Amidst these two thresholds, the systems is in a sort of *gray zone* where specific simulations are needed to determine the optimal α. For energy values in the order of magnitude of the considered microcontroller the policy shows an almost stepwise behavior: α is always 0 when the average of the processes' periods is below 7s, while α is 1 when the average of the processes' periods is above 11s. In the gray zone between 7s and 11s simulations have identified intermediate values of α ranging from 0.3 to 0.7, especially around the critical value of 8s. In such cases, by using the correct α parameter to tune the policy, energy savings up to 20% can be obtained. A similar trend can be observed for $2\times$ and $4\times$ perturbations, while for perturbations greater than $8\times$, which however account for extreme cases, the *gray zone* is more extended. Finally, we studied and estimated the energy gain of the hibernation approach (HA) with respect to classical operation (CO), in which the system is brought in sleep mode during idle periods. The executions periods have been varied between 4 and 250 seconds, while the process image sizes was selected randomly between between 1 and 32 KB. The energy gain in function of the average period rises from the 8%, with average periods of 10s, to the 53% for average periods of 20s to the 84% for periods of 50s and steadily over the 90% for average periods over the 100s. The energy gain with respect to the average size of the swap-memory varies from over the 90%, for sizes in the range of 1 KB, to about the 70% for average sizes of 6 KB to more than the 50% for heavy sizes, in the order of 12KB.

VIII. Conclusions

This paper presented the achievements obtained by our research on a smart autonomous hibernation mechanism and an optimal hibernation policy enabling energy-efficient operation in high and mid-range WSN nodes. Results shows energy gains up to 50% compared to widely used trivial hibernation policies, as well as a good robustness and stability.

References

[1] Brandolese,C., Fornaciari,W. Rucco,L. and Terraneo, F. Enabling ultra-low power operation in high-end wireless sensor networks nodes. In Proc. of CODES+ISSS '12. ACM, New York, NY, USA, 433-442.2012.

[2] Estrin, D. and Girod, L. and Pottie, G. and Srivastava, M., Instrumenting the world with wireless sensor networks, *In Proc. of ICASSP '01*, pp. 2033–2036 vol.4, 2001.

[3] Estrin, D., Tutorial on Wireless Sensor Networks, *In MobiCom'02*, Atlanta, Georgia, USA, 2002.

[4] Anastasi,G., Conti,M., Di Francesco,M. and Passarella,A. Energy conservation in wireless sensor networks: A survey. Ad Hoc Netw. 7, 3 (May 2009), 537-568. 2009.

[5] Brandolese,C., Fornaciari,W. Rucco,L., Hibernation Model for Energy Optimization of Multi-Tasking Wireless Sensor Networks, Tech. Rep. n. 2103.4 Politecnico di Milano - DEIB, Italy.