On the Monitoring of System-Level Energy Consumption of Battery-Powered Embedded Systems

Arliones Hoeller Jr.
Automation and Systems Department
Federal University of Santa Catarina
Florianópolis, Brazil
arliones@das.ufsc.br

Antônio Augusto Fröhlich
Software/Hardware Integration Lab
Federal University of Santa Catarina
Florianópolis, Brazil
guto@lisha.ufsc.br

***Abstract*—This paper addresses an approach for accurately measuring energy consumption on battery-powered embedded systems which can be adequately tuned in order to enhance a set of timing and energy consumption metrics for mission critical systems. We introduce a software-based accounting scheme which is calibrated by low-precision battery state-of-charge reads through a battery voltage model. We then perform an offline multi-objective optimization procedure using NSGA-II to find good candidates to the period at which battery consumption information should be updated. Such candidates might guarantee timing constraints (i.e., no deadline misses), minimize residual energy after a pre-defined system lifetime, and maximize system utilization. We considered a simple scheduler which will reserve battery charge to run hard real-time tasks during a pre-defined lifetime and will prevent best-effort tasks from running whenever accounted battery state-of-charge is bellow the current reserve. We evaluated our approach by performing case-studies.**

***Index Terms*—accounting, energy, embedded systems, real-time scheduling, adaptive task periods**

I. Introduction

Low energy consumption is an important non-functional requirement for the design of battery-powered embedded systems. Reliable information on the system energy source is of paramount importance to design an energy-efficient system. In order to keep track of exact energy consumption at runtime, traditional approaches rely on continuous measurements of the amount of current drained from the battery. Besides the additional hardware required to perform this task, software support for sampling such circuitry may compromise system performance due to the requirement of fine grained information needed to sample this continuous signal.

To cope with this requirement mobile systems provide means to measure battery voltage by which it is possible to infer battery charge through an approximate discharge model for a given battery. This approach, however, brings limitations to the task of estimating battery charge. Among the reported problems [2], [3] there is one of special interest for the task of precisely estimating battery charge on embedded systems: the low accuracy and long response time of voltage-based battery state-of-charge models. This problem is related to the diminished precision of such voltage measurements implied by low resolution analog-to-digital converters and to the oscillation on the battery voltage-levels due to load variations. It can be easily illustrated by Fig. 1. The figure shows that during most of system lifetime the response time in terms of battery voltage varies greatly. For instance, if an energy-related decision lowers system utilization on day 50, this decision may last until the next expected voltage drop, around day 100, making an unique decision stay in effect for 14% of system lifetime. We can also see that the scheduler misses the opportunity of raising system utilization because its monitor is not able to rapidly inform that the system is not using the expected amount of energy. Instead of gradually raising system utilization and keeping it higher during a longer period of time, it only acts by the end of a given period, causing bursts in system utilization for short periods of time.

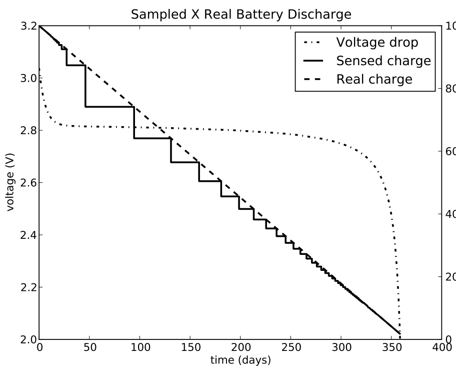


Fig. 1. Sampled discharge plotted against real discharge for a CR2 series Panasonic lithium battery [1].

In this paper we propose a software-based energy consumption accounting scheme. We rely on the fact that energy consumption is a cross-cutting concern, orthogonal to all system components [4], to enable accurate monitoring of energy consumption in a component-based operating system for embedded systems. This scheme monitors the execution of operating system functionalities. In order to make this approach feasible, we derived three different profiles for energy accounting which can be applied to devices with different operational behavior. We evaluate the effect of our approach on system performance. This paper focus exclusively in the

This work have been partially funded by CNPq grant 479009/2009-0. Arliones Hoeller Jr is also supported in part by CAPES grant RH-TVD 006/2008.

proposed model. An implementation of the system was also performed for the EPOS platform [5] by extending EPOS' power management scheme [6], altought the implementation details where kept out of this paper due to length limitations.

The remaining of this paper is organized as follows. Section II presents an overview of related work. Section III describes the software-based energy accounting approach proposed in this paper, analyzing it through a didactic case-study. Section IV presents a case-study of the proposed approach on a wireless sensor network system. Section V summarizes this paper and gives some insights on future work.

II. RELATED WORK

Related work were divided into those related to software-based accounting of energy usage and those which address the deployment of meta-heuristic and multi-objective optimization methods on energy-aware real-time systems.

A. Accounting of Battery Usage

We investigated similar battery monitoring schemes in operating systems for wireless sensor networks, including TINYOS [7], MANTISOS [8] and CONTIKI [9]. All three systems provide access to battery voltage measurements through an ADC interface. None of them provide complete voltage models by which it would be possible to infer battery state-of-charge from voltage levels.

Yang et al. [10] built an extension to TINYOS that enables software-based accounting of energy consumption and battery lifetime estimation for the MICA2 sensor node. They monitor the time each hardware component stays in an operating mode by intercepting mode changes and accumulating drawn current during this period, decrementing it from the initially informed battery charge. Their approach, however, requires modifications on every monitored system component and may pose unnecessary overheads in situations where devices change operating mode too often, as it would be the case of a radio transceiver in a low-power listen mode, where the transceiver is periodically switched on and off to check for incoming messages. Weissel and Kellner [11] used an event-based energy accounting mechanism to compute energy on the BTNODE platform [12] running TINYOS. They, however, didn't implement the presented concepts, limiting their work to the analysis of the implementation possibilities.

Dunkels et al. [13] implemented a time-based energy accounting system for CONTIKI running on TMOTE SKY [7] platform. Their model, as does the approaches used in TINYOS, demand for modification in several different operating system modules (i.e., drivers), what may make the system difficult to maintain. They also show that the lack of calibration with real information in their system may be the cause of significant errors in energy estimations.

B. Optimization Methods in Energy-Aware Real-Time Systems

Energy optimization for real-time systems has long been a subject of great interest in the real-time community [14]. A plurality of works have been published that apply optimization techniques on real-time system models to find good tradeoffs between energy consumption and operating frequency/voltage of CPUs (DVS - Dynamic Voltage Scaling) [15], on/off status of peripheral devices (DPM - Dynamic Power Management), or both [16]. All these works, although important to the design of energy-aware real-time systems, are outside the scope of this paper for they are orthogonal to the work presented here.

Chantem et al. [17] proposed a generalized elastic scheduling framework for real-time tasks based on Buttazzo's elastic model [18] which may adapt task's elastic periods online based on one specific (generic) performance metric. Optimal period adjustments are then performed by a heuristic proposed by them. Although energy-related metrics may be used as the optimization objective, authors didn't explore this. Eker et al. [19] and Cervin et al. [20] show the application of optimization theory to solve the period selection problem at runtime by performing adaptive adjustments of periods based on a control performance metric. Bini and Natale [21] devised an optimal search algorithm to minimize tasks' frequencies by performing incremental improvements on one specific performance metric by using a branch and bound search over a predefined feasibility region of the domain of task frequencies until the global optimum is reached. The algorithm applies to fixed-priority scheduling schemes and may be only applicable offline due to its high complexity.

To the best of our knowledge, no work explored the effects of the period at which battery state-of-charge information is made available for a real-time energy-aware scheduler.

III. BATTERY LEVEL MONITORING BY EVENT ACCOUNTING

Traditional voltage-based battery monitoring may lead to a pessimistic bias of an energy-aware task scheduler. In order to enhance the precision of the battery monitor, we propose a software-based scheme to account for energy consumption. This scheme is based on the premise that energy is consumed by hardware, not software, but it is the software that controls and monitors hardware activity. Considering that the operating system layer abstracts hardware access to application, it is straightforward to assume that the operating system is the entity with most knowledge about hardware activity, thus being able to monitor system-wide energy consumption.

A. Energy consumption profiles

We analyzed usual hardware behavior and modeled three different profiles to account for energy consumption: time-based measurement, event-based measurement, and combined measurement. The time-based profile is used to account for energy consumption of devices draining constant current over time when in a specific operating mode, as (1) shows. The event-based profile is used in devices for which operation can be mapped to specific events (e.g., sensor sampling). As shown in (2), events are accounted for and energy is updated periodically based on this accounting. In some devices, however, both approaches may be used. For instance, a radio that stays in a low-power listen mode has a base energy

TABLE I
PROCESSING OVERHEAD OF THE ENERGY ACCOUNTER ON AN ARM7-TDMI PROCESSOR.

Procedure	Time overhead
migration_update_energy	73 cycles
event_update_energy (1 repetition)	29 cycles
event_update_energy (n repetitions)	63 cycles

consumption (computed by the time-based profile) and extra energy consumption when data actually arrives (computed by the event-based profile). As shown in (3).

$$E_{tm}(dev) = (t_{end} - t_{begin}) \times I_{dev,mode} \quad (1)$$

$$E_{ev}(dev) = \sum_{event\_counters} E_i * counter \quad (2)$$

$$E_{tot}(dev) = E_{tm}(dev) + E_{ev}(dev) \quad (3)$$

, where $E_{tm}(dev)$ is the energy consumption in the time-based profile for a specific device, t_{end} and t_{begin} denote timestamps, and I is the current of dev at a given $mode$. E_{ev} denotes the sum of energy (E_i) consumed by the observed events ($counters$). E_{tot} is the energy consumption in the combined profile.

B. Battery state-of-charge monitoring

At runtime, battery charge is updated with the accounted data of each device. These accounting information, however, need to be collected in order to update battery charge. The frequency in which these data are collected directly affects the accuracy of the proposed battery state-of-charge monitor. Recalling the curves in Fig. 1, we may say that high update frequencies would draw a curve close to the "real charge", while low update frequencies would approximate the "sensed charge" curve. In this section we describe how to achieve a satisfactory curve close to the "real charge" curve of Fig. 1 by adequately adjusting the battery update frequency.

We start by analyzing the update frequency for the time-based profile. It is not the intent of the present work to investigate issues related to the frequency of operating mode migrations or their time and energy overheads, for such problems have already been extensively addressed [22], [23]. Thus, this profile can adhere to such migration models by the inclusion of an extra routine like $energy\_migration\_update$ on Fig. 2 to compute elapsed time and consumed energy during these migrations as described by (1). The execution time for this routine, shown in Tab. I is constant and can be easily obtained and integrated to any transition model, being either real-time or not. Additionally, to prevent the system from loosing control of battery discharge when devices stay in a certain operating mode for long periods, an active component periodically collects energy consumption information from all devices and updates battery charge.

For the event-based profile, however, the battery charge update approach needs to be different to avoid unnecessary processing overheads. For instance, suppose that a hypothetic system monitors an event that is the reception of a byte from a

```
procedure energy_migration_update()
    now = SystemTime();
    Battery −= (now − time_begin) ∗ I[mode]; // (1)
    time_begin = now;
end procedure;

procedure energy_event_update(event)
    Battery −= E[event] ∗ C[event]; // (2)
    C[event] = 0;
end procedure;

procedure energy_update_total()
    for each device do
        for each event of device do
            energy_event_update(event); // (3)
        energy_migration_update();
    end for;
    Battery_Charge = from battery voltage model;
    Battery = max(Battery_Voltage, Battery); // (4)
end procedure;
```

Fig. 2. Algorithms for energy accounting.

network interface. Network protocols will seldom use only one byte to perform communications, thus, as can be seen in Tab. I, system performance may benefit from periodic updates of an accumulated counter. In order to do that, an active object was modeled as an extra task on the system which is responsible for collecting accounted information of the event-based profile.

It is important to note that the accounting mechanism employed in this scheme, although more accurate, is still pessimistic once it is based on the worst-case energy consumption (WCEC) of events and components' operating modes. Thus, it is expected that the accounted energy consumption reaches the value read from the battery voltage model before it shows a drop in voltage. It is safe, however, to assume that the information from the voltage model is a secure bound to battery charge, although conservative. Then, we may correct the battery charge to the maximum value between the accounted charge and the one estimated by the voltage model (as shown in (4)).

$$E_{batt} = max\left(E_{volt}, E_{batt} - \sum_{i=0}^{\#devs} E_{tot}(i)\right) \quad (4)$$

Finally, the active component with the task of periodically updating the battery charge is responsible for collecting accounted information from both event-based and time-based profiles. The algorithm is the one at the procedure $energy\_update\_total$ of Fig. 2. It is important to note that timestamps and event counters are reset every time energy accounting is updated (by $energy\_migration\_update$ and $energy\_event\_update$), thus making sure that no energy consumed is accounted for twice. This algorithmic approach assumes initialization of $Battery$ with the nominal capacity of the battery in use.

C. On the Freshness of Battery Information

To understand the accounter behavior further we performed an exploration of system design space to be able to determine

TABLE II
HYPOTHETIC APPLICATION TASKS' PARAMETERS[1].

T	P	WCET	WCEC	1-hour
H_1	100	25	1.0	0.0036
H_2	150	25	1.4	0.00336
H_C[2]	50	0.0000453	0.000004156	0.00000299
B_1	200	50	2.0	0.0036
Energy consumption of mandatory tasks				0.0069603

the frequency at which accounted data should be gathered provided that the system has a pre-defined requirement of operation lifetime. We used a multi-objective optimization method based on the *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) [24] which is able to find good solution candidates for the frequency of the collector task, i.e., those closer to the Pareto front. The optimization is executed with two objectives: to minimize residual energy and to maximize the execution rate of best-effort tasks. We use a simple scheduling mechanism in which hard real-time tasks execute regardless of system energy availability and best-effort tasks run only when the system is still able to guarantee energy availability for hard real-time tasks. Priorities in the scheduling queue are assigned through a Rate Monotonic policy provided that all hard real-time tasks have higher priorities than any best-effort task, regardless of their period. We also assume that initial battery charge is enough to guarantee hard real-time tasks' executions during the expected lifetime.

Now we analyze a hypothetic application. Tab. II shows the parameters for this application, comprised by two hard real-time tasks (H_1 and H_2), one best-effort task (B_1) and the energy accounter collector task (H_C), ordered according to their priority. We kept the collector task as a hard real-time task for two reasons. First, it is the only way to guarantee that the defined period for the collector task will be respected, once best-effort tasks may be prevented from executing. Second, if eventually the system stops the execution of best-effort tasks and the collector task is a best-effort task, the battery information will no longer be updated, thus being this a dead-end for the energy-aware scheduler.

In order to evaluate the system instances (individuals) generated during the optimization process we integrated a real-time simulator to the optimizer. This system simulates a rate monotonic queue with two levels of priorities, being the first one the task's class and the second one the rate monotonic priority itself. This made it possible the separation between hard real-time and best-effort tasks. The simulator also monitors energy consumption of tasks and controls battery discharge based on informed worst-case energy consumption (WCEC) of tasks. In order to achieve a more realistic behavior we consider that all tasks actually use their WCEC for 75% of the jobs. The remaining 25% of the jobs have a random energy consumption uniformly distributed between 50% and

[1]T: task; P: period in ms; WCET: worst-case execution time in ms; WCEC: worst-case energy consumption in ηAh; 1-hour: energy consumption for the targeted lifetime (1 hour) in mAh.

[2]This is a worst-case scenario as values of "P" and, as consequence, "1-hour", are to be defined by the optimization.

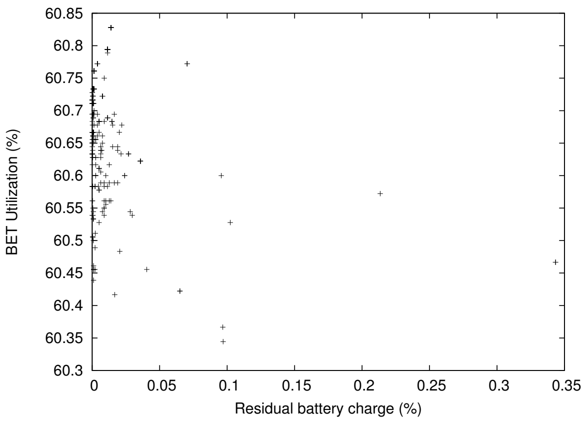


Fig. 3. All solutions for the hypothetical application.

TABLE III
SOLUTIONS FOR THE FREQUENCY OF THE COLLECTOR TASK FOR THE HYPOTHETICAL APPLICATION.

Frequency (Phenotype)	Period (s)	Best-effort rate	Remaining Charge
11.790 Hz	0.085	60.794%	0.114 ‰
11.793 Hz	0.085	60.761%	0.014 ‰
13.089 Hz	0.076	60.728%	0.001 ‰
13.161 Hz	0.076	60.717%	0.001 ‰
14.468 Hz	0.069	60.828%	0.139 ‰
15.812 Hz	0.063	60.772%	0.039 ‰
16.455 Hz	0.061	60.733%	0.007 ‰

100% of the WCEC. This generates a slack on the energy budget that can be used by the best-effort tasks, as would actually happen on real systems. Although naive, this simple probabilistic assumption helps to understand and analyze the approach. A more consistent approach will be considered later in the case study of Section IV.

We ran NSGA-II with a population size of 100 individuals, being 20 of them selected as parents, generating 20 offsprings, repeating the process during 50 generations (iterations) ($\alpha = 100, \mu = 20, \lambda = 20, g = 50$). The best solutions found by the optimizer, i.e., those at the Pareto front, are shown in Tab. III. By analyzing Fig. 3, which shows all found solutions, it is possible to observe here the wide spread of results obtained from the optimization process. It is also important to note the non-linear behavior of the observed parameters in relation to frequency, showing why the solution to this problem benefits from the application of meta-heuristic methods like NSGA-II.

IV. EVALUATION

In this section we present the deployment of the approach described in this paper in a mobility-enabled wireless sensor network running the Ant-based Dynamic Hop Optimization Protocol (ADHOP) over an IP network using IEEE 802.15.4. ADHOP is a self-configuring, reactive routing protocol inspired by the HOPNET protocol for *Mobile Ad Hoc Networks* (MANETs) and designed with the typical limitations of sensor nodes in mind, energy in particular [25]. ADHOP's reactive component relies on an *Ant Colony Optimization* algorithm to discover and maintain routes. Ants are sent out to

TABLE IV
ADHOP CASE-STUDY TASKS' PARAMETERS[3].

T	P	WCET	WCEC	25-days
$Sense$	$1,000$	2	10.584	441.50
$Forward$	$1,000$	50	268.056	378.43
$Collector$[4]	5	0.0000453	0.000004156	0.0017954
LPL	5	0.25	0.000001111	0.00048
$Route$[5]	500	100	490.278	$2,118$
Energy consumption of mandatory tasks				601.88

TABLE V
SOLUTIONS FOR THE FREQUENCY OF THE COLLECTOR TASK FOR THE ADHOP CASE-STUDY.

Frequency (Phenotype)	Period (s)	Best-effort rate	Remaining Charge
23.944 Hz	0.042	9.92%	0.008 ‰
41.243 Hz	0.024	9.93%	0.130 ‰
41.849 Hz	0.024	9.93%	0.013 ‰

track routes, leaving a trail of pheromone on their way back. Routes with a higher pheromone deposit are preferred for data exchange.

With the purpose of corroborating the approach presented in this paper we modified ADHOP in order to make it energy-aware. We classified ADHOP's tasks as mandatory (hard real-time) or optional (best-effort). The main idea behind this setup was to homogenize the battery discharge for every node in the network to enhance the lifetime of the network as a whole. Considering the radio the most energy-hungry component in a wireless sensing node, we toke the design decision of modeling the routing activity of ADHOP as best-effort tasks, as shown by the task set at Tab. IV. The basic node functionality of sensing a value (task $Sense$) and forwarding it through the radio to the next node (task $Transmit$) where modeled as hard real-time tasks. The functionality of forwarding other nodes' packets (and ants) when acting as a "router" was modeled as two best-effort tasks, one for monitoring the channel for arriving messages (LPL - Low Power Listen), and another to effectively receive the message and route it to another node ($Route$).

We set the lifetime objective for this system to 25 days. By analyzing the task set it is possible to compute the total energy consumption of hard real-time tasks for the desired lifetime to be of $601.88mAh$, thus, the initial battery charge for the system has to be greater than that. We then defined the system battery as an of-the-shelf CR-2/3V battery with a total capacity of $850mAh$.

We simulated several network scenarios (with up to 200 nodes) using the *Global Mobile Information System Simulator* (GLOMOSIM). In this setup, nodes were programmed to communicate intensively and move randomly within a grid of 700 x 400 meters for 25 days, thus stimulating both the routing protocol and the power management mechanisms. GLOMOSIM was integrated to the same NSGA-II optimizer described in Section III-C, and the optimization process ran with the same parameters. The results of this optimization are shown in Tab. V.

Also, we analyze the impact on the network performance by comparing the obtained results with the data originally published by Okazaki [25]. Fig. 4 (above) shows a reduction on the average energy consumed by each node on the network while Fig. 4 (bellow) shows the expected enhancement on the average battery lifetime of nodes. It is important to note that all nodes in the network lived for, at least, 25 days as expected, being that the reason why the average lifetime stayed well above it, around 30.

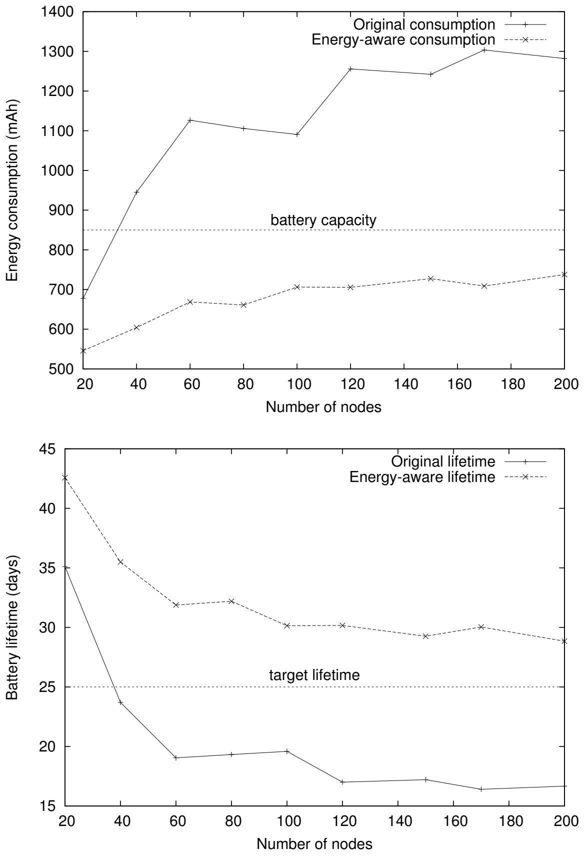


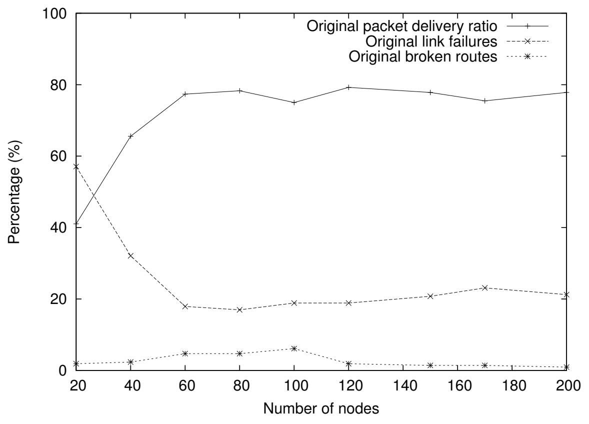
Fig. 4. Average energy-related parameters for the simulated ADHOP setup.

Besides the good results from the energy consumption perspective, we observed an important decrease on the overall network quality, as shown in Fig. 5 for the "Broken routes" and "Delivery ratio" parameters. These contrast, however, with the obtained results on "Link failures", indicating that ADHOP deals well with the broken routes, allowing undelivered packets to be re-routed and finally delivered.

[3]T: task; P: period in ms; WCET: worst-case execution time in ms; WCEC: worst-case energy consumption in ηAh; 25-days: energy consumption for the targeted lifetime (25 days) in mAh.

[4]This is a worst-case scenario as values of "P" and, as consequence, "25-days", are to be defined by the optimization.

[5]"Route" is an sporadic task. Once it is a best-effort task in the system, we consider a hypothetic frequency of 2 Hz (period of 500 ms) to show the impact of routing in the node energy consumption.



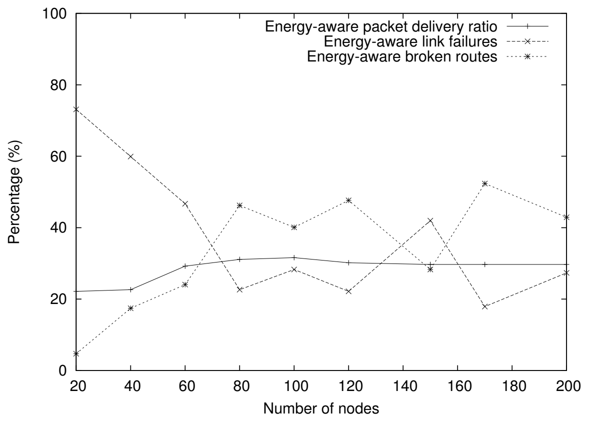


Fig. 5. Impact on network quality for the ADHOP case-study.

V. Conclusion

We presented a software implementation of an energy consumption accounter for battery-operated embedded systems. We modeled the accounter and implemented it in a simulation environment and in a real platform (EPOS [5]), altought the implementation details in the real platform have been kept out due to the page limit. In order to lower the processing overhead imposed by our approach we extracted runtime parameters of energy consumption and execution time of a given application and submitted it to a meta-heuristic optimizer (NSGA-II). The objective is to look for good solutions for the period at which battery-related information should be updated in order to maximize system utilization while minimizing residual energy and guaranteeing a pre-defined system lifetime (mission duration). A case study on an IP-based network running over IEEE 802.15.4 sensing nodes showed promising results. It showed that the approach was able to control energy consumption on the network in a way that none of the nodes ran out of battery before the pre-defined mission duration has elapsed.

On going studies are extending the work in this paper by deploying fairer scheduling mechanisms to reduce the impact on system quality. This effort rely on flexible task scheduling schemes to be put in place of the current egoist approach of preventing tasks' execution and using network-wide battery charge information provided by the accounter described herein as parameter for ADHOP's pheromone generation function.

References

[1] Panasonic, *Manganese dioxide lithium batteries datasheet*, Japan, 2006.

[2] T. Mundra and A. Kumar, "Micropower battery state-of-charge monitor," *Consumer Electronics, IEEE Trans. on*, vol. 54, no. 2, May 2008.

[3] M. Penella and M. Gasulla, "Runtime extension of low-power wireless sensor nodes using hybrid-storage units," *Instrumentation and Measurement, IEEE Transactions on*, vol. 59, no. 4, pp. 857 –865, Apr. 2010.

[4] D. Lohmann, W. Schroder-Preikschat, and O. Spinczyk, "Functional and non-functional properties in a family of embedded operating systems," in *Object-Oriented Real-Time Dependable Systems, 2005. WORDS 2005. 10th IEEE International Workshop on*, feb 2005, pp. 413 – 420.

[5] LISHA, "Epos project website," Internet, feb 2011. [Online]. Available: http://epos.lisha.ufsc.br

[6] A. A. Frohlich, "A comprehensive approach to power management in embedded systems," *Hindawi Int. Journal of Distributed Sensor Networks*, vol. 2011, p. 19, 2011, article ID 807091. [Online]. Available: http://www.hindawi.com/journals/ijdsn/aip/807091/

[7] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *4th Int. Conference on Information Processing in Sensor Networks*, Los Angeles, USA, Apr. 2005.

[8] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han, "Mantis os: an embedded multithreaded operating system for wireless micro sensor platforms," *Mob. Netw. Appl.*, vol. 10, pp. 563–579, August 2005.

[9] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Proc. of the 1st IEEE Works. on Embedded Networked Sensors*, Tampa, USA, Nov. 2004.

[10] T. Yang, Y. K. Toh, and L. Xie, "Run-time monitoring of energy consumption in wireless sensor networks," in *IEEE Int. Conference on Control and Automation*, Jun. 2007, pp. 1360 –1365.

[11] A. Weissel and S. Kellner, "Energy-aware reconfiguration of sensor nodes," in *Proc. of the 1st GI/ITG Works. on Non-Functional Properties of Embedded Systems*, Nuremberg, Germany, mar 2006, pp. 69–75.

[12] J. Beutel and et al., "Prototyping wireless sensor network applications with btnodes," in *Wireless Sensor Networks*, ser. LNCS. Springer, 2004, vol. 2920, pp. 323–338.

[13] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *Proc. of the 4th Workshop on Embedded Networked Sensors*, Cork, Ireland, jun 2007.

[14] L.-C. Weng, X. Wang, and B. Liu, "A survey of dynamic power optimization techniques," in *Proc. of the 3rd IEEE Int. Workshop on System-on-Chip for Real-Time Applications*, jul 2003, pp. 48 – 52.

[15] J.-J. Chen and C.-F. Kuo, "Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms," in *13th IEEE Int. Conference on Embedded and Real-Time Computing Systems and Applications*, aug. 2007, pp. 28 –38.

[16] N. Jha, "Low power system scheduling and synthesis," in *IEEE/ACM Int. Conference on Computer Aided Design*, 2001, pp. 259 –263.

[17] T. Chantem and et al., "Generalized elastic scheduling for realtime tasks," *IEEE Trans. on Computers*, vol. 58, no. 4, pp. 480–495, apr 2009.

[18] G. Buttazzo and et al., "Elastic task model for adaptive rate control," in *Proc. of the 19th IEEE Real-Time Systems Symp.*, dec 1998, pp. 286–295.

[19] J. Eker and et al., "A feedback scheduler for real-time controller tasks," *Control Engineering Practice*, vol. 8, no. 12, pp. 1369–1378, 2000.

[20] A. Cervin and et al., "Feedback–feedforward scheduling of control tasks," *Real-Time Systems*, vol. 23, pp. 25–53, 2002.

[21] E. Bini and M. Di Natale, "Optimal task rate selection in fixed priority systems," in *26th IEEE Int. RealTime Systems Symp.*, dec 2005, p. 11.

[22] A. J. Hoeller, L. F. Wanner, and A. A. Fröhlich, "A Hierarchical Approach For Power Management on Mobile Embedded Systems," in *5th IFIP Working Conf. on Distributed and Parallel Embedded Systems*, Braga, Portugal, Oct. 2006, pp. 265–274.

[23] E. Seo, S. Kim, S. Park, and J. Lee, "Dynamic alteration schemes of real-time schedules for i/o device energy efficiency," *ACM Trans. on Embedded Computing Systems*, vol. 10, pp. 23:1–23:32, January 2011.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 2, pp. 182 –197, Apr. 2002.

[25] A. M. Okazaki and A. A. Fröhlich, "Ad-zrp: And-based routing algorithm for dynamic wireless sensor networks," in *Proc. of the 18th IEEE Int. Conference on Telecommunications*, Ayia Napa, Cyprus, may 2011.