# RT scheduling in energy-harvesting WSN

| | |
|---|---|
| Journal: | *Transactions on Embedded Computing Systems* |
| Manuscript ID: | Draft |
| Manuscript Type: | Paper |
| Date Submitted by the Author: | n/a |
| Complete List of Authors: | Hoeller Jr., Arliones; UFSC, DAS<br>Fröhlich, Antônio Augusto; UFSC, LISHA |
| Computing Classification Systems: | D.4.7 - Real-time systems and embedded systems |
| | |

# Scheduling of energy usage for real-time wireless sensor network systems with the energy-harvesting capability

ARLIONES HOELLER JR., Federal University of Santa Catarina
ANTÔNIO AUGUSTO FRÖHLICH, Federal University of Santa Catarina

Wireless sensor network (WSN) systems present serious limitations in terms of energy availability and processing power. To cope with the energy availability issue, WSN nodes equip energy harvesters (e.g. photovoltaic modules) and energy buffers (e.g. rechargeable batteries). This approach, besides increasing energy availability, also brings uncertainty to the system when the efficiency of the energy harvester depends on variables that are difficult to predict (e.g. solar irradiation, temperature). This paper presents a set of heuristics to schedule energy utilization of critical applications running on wireless sensor networks. Experiments consider the impact of the scheduling heuristics on timing correctness and network quality. We propose and evaluate four energy scheduling heuristics based on the recent history of harvested energy. Results of simulations based on a real system implementation show the performance of the heuristics.

Categories and Subject Descriptors: D.4.7 [**Organization and Design**]: Real-time systems and embedded systems

General Terms: Design, Experimentation, Performance

Additional Key Words and Phrases: wireless sensor networks, energy harvesting, real-time scheduling

**ACM Reference Format:**
Hoeller Jr., A., Fröhlich, A.A. 2013. Scheduling of energy usage for real-time wireless sensor network systems with energy-harvesting capability. ACM Trans. Embedd. Comput. Syst. 12, 4, Article 1 (September 2013), 18 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

## 1. INTRODUCTION

The energy consumption of devices in a wireless sensor network (WSN) is a serious concern for system designers. These systems usually need to operate for long periods of time without having their energy sources replaced or recharged. Also, *long lime* is a relative concept, once required operation time will always depend on several variables, related to both the environment and the application itself. Besides the operation time, complexity of WSN systems is increasing as deployments begin to spread to a variety of scenarios. Such scenarios often present critical requirements where timing and correctness also become crucial issues. At this point, energy and correctness become conflicting requirements, once traditional means to increase correctness imply on higher energy consumption and means to reduce energy consumption often degrade system performance.

This work is supported in part by Brazilian RNP/CTIC effort towards the *Internet of Things*. A. Hoeller Jr. is also supported in part by CAPES, under grant RH-TVD 006/2008.
Author's addresses: Arliones Hoeller Jr., Department of Automation and Systems Engineering, Federal University of Santa Catarina; Antônio Augusto Fröhlich, Department of Computer Science, Federal University of Santa Catarina.

The idea of extracting energy from the environment has become an attractive alternative to deal with the energy deficit and extend the longevity of sensor nodes. Although the energy harvesters bring more energy into the system, the extra energy needs to be intelligently stored and used to stabilize system performance. This happens because the energy input of these harvesters varies according to the physical phenomena used to produce electric energy. For instance, energy production in systems using solar or wind as energy sources is subject to weather conditions that vary seasonally and for which accurate prediction methods are not available [Fay and Ringwood 2010]. Thus, the systems use energy-aware schedulers to adapt the system behavior in accordance to energy availability and consumption. In the context of real-time systems, the successful operation of a system depends on the compliance to the requirements of its critical parts. As a consequence, an energy-aware scheduler for real-time systems must somehow guarantee that there will be enough energy to execute the critical parts of the system.

This work presents an adaptive real-time scheduling method for systems under oscillatory energy availability. The approach includes mechanisms to ensure the scheduling of hard real-time tasks by guaranteeing their time constraints and their energy demands for a previously defined period. Guarantees for hard real-time tasks rely on the worst-case computation time and worst-case energy consumption of each task. At runtime, the system scheduler uses slack time and surplus energy to execute a set of best-effort tasks. The scheduling of best-effort tasks uses heuristics to get the best benefit from the incoming energy while trying to stabilize the rate of execution of these tasks.

A case-study presents the use of the proposed approach in a mobile WSN application that presents both hard real-time and best-effort tasks. In the application, the mandatory functionalities of sensing data and forwarding it through the network are hard real-time tasks. Once the routing functionality often consumes a lot of energy in mobile sensor networks, a set of best-effort tasks implements the routing mechanism. This means that routing only takes place if there is enough energy. The only entity that manages power in the system is the energy scheduler proposed here. In the experiments, one instance of the energy scheduler operates independently in each node of the network, making its energy management decisions solely based on energy-related information of the node.

The paper structure is as follows. Section 2 details the task model, including the scheduling approach, and the energy model, including monitoring of energy consumption, production, and storage. Section 4 presents the approach for adapting system energy consumption in order to deliver hard real-time guarantees and maximize the execution rate of best-effort tasks. Section 5 presents a WSN case study where routing functions are best-effort tasks and nodes harvest energy from photovoltaic panels. Section 6 presents related work. Finally, Section 7 closes the paper.

## 2. SYSTEM MODELS

The definition of a system model allows a better understanding of the problem in hand. This section presents two models: the *Task Model* and the *Energy Model*. The first models the system quality and timing correctness issues. The second, models aspects of energy consumption, production, and storage.

### 2.1. Task Model

Real-time scheduling is a common solution to ensure correctness and quality of critical applications. In a hard real-time model, all instances of system tasks (jobs) must complete in time. Flexible real-time models may also take into account a set of optional or adaptable tasks. These flexible models may also be called best-effort models. In this scenario, the real-time scheduling policy ensures correctness, while system quality levels may be expressed as a function of the execution rate or adaptation levels of a set of flexible tasks. This work employs a hybrid model based on both hard real-time and best-effort tasks.

As a matter of fact, critical real-time systems are almost always designed considering energy sources that are consistent with system demands. Also, the system designer must make energy saving decisions, such as voltage scaling and device hibernation, at design-time. Thus, the designer can take such decisions into consideration while defining the energy budget necessary to sustain the system. Complex, battery-operated, real-time embedded systems, such as satellites, autonomous vehicles, and even sensor networks, comprise a set of tasks that include both, critical and noncritical tasks. A power manager for one such embedded system must follow the design-time decisions for critical parts while trying to optimize energy consumption by noncritical parts.

For the envisioned scenario – i.e. battery-operated, real-time, embedded systems harvesting energy from the environment – the energy budget would be defined at design-time based on critical tasks, while noncritical tasks would be executed on a best-effort policy, considering not only the availability of time, but also of energy. Along with the assumption that an autonomous power manager cannot interfere with the execution of hard real-time tasks, the separation of critical and noncritical tasks at design-time leads to the following scheduling strategy:

— The system handles hard real-time tasks as mandatory tasks, executed independently of the energy available at the moment. These tasks are scheduled according to traditional algorithms such as Earliest Deadline First (EDF) and Rate Monotonic (RM) [Liu and Layland 1973], either in their original form or extended to support DVS [Pillai and Shin 2001].
— Best-effort tasks, periodic or not, have lower priorities than the hard real-time ones and therefore only execute if no hard real-time tasks are ready to run. Furthermore, the decision to dispatch a best-effort task must also take into consideration whether the remaining energy will be enough to schedule all activations of hard-real time tasks expected to occur until the system reaches the targeted lifetime.
— During system execution, a speculative power manager is active identifying idle periods of components. Whenever a best-effort job does not execute due to energy limitations, the number of idle periods of system components raises, thus promoting further energy savings.

This scheduling strategy has only small implications in terms of process management at the operating system level; however, in order to be implemented it requires a comprehensive power management infrastructure, like the one previously presented by the authors [Fröhlich 2011; Hoeller Jr. and Fröhlich 2011]. In particular, the system needs battery monitoring services to support the scheduling decisions around best-effort tasks and component dependency maps to avoid power management decisions that could affect the execution of hard real-time tasks.

## 2.2. Energy Model

In this work, three facets of energy must be taken into account: consumption, production, and storage.

Equations 1 through 3, introduced in a previous work [Hoeller Jr. and Fröhlich 2011], define the employed energy consumption model. Depending on the behavior of a given component, or the information available about its energy consumption, the designer may choose to monitor energy consumption based either on the time the device spends in a particular operating mode or on the events generated by the device. $E^i_{tm}(d, \phi)$ defines the energy consumed by a single device ($d$) over time as a function of current drain ($I$) and time ($t$) spent in an operating mode ($m$) with a specific configuration ($\phi$). $E^i_{ev}(d, \phi)$ is the sum of the energy consumed by events that are relevant in terms of energy consumption. During execution, the system accounts for these events ($\chi$). Each event has a known worst-case energy consumption ($E_e(\phi)$) that is subject to a specific configuration ($\phi$). In the case of a device consuming energy in both ways, both energy consumption profiles can be applied. Finally, $E^i_{tot}(\phi)$ is the sum of the estimated energy consumption of all system devices ($\Delta$), during the $ith$ iteration.

$$E^i_{tm}(d, \phi) = (t_{end} - t_{begin}) \times I(d, m, \phi) \tag{1}$$

$$E^i_{ev}(d, \phi) = \sum_{e=0}^{|\chi|} (E_e(\phi) \times \chi_e) \tag{2}$$

$$E^i_{tot}(\phi) = \sum_{d=0}^{|\Delta|} (E^i_{tm}(d, \phi) + E^i_{ev}(d, \phi)) \tag{3}$$

$E^i_{tm}$ is updated either on every operating mode change or periodically, when $E^i_{ev}$ and $E^i_{tot}$ are also updated. The period of the iterations ($i$) vary from application to application and has already been subject of previous studies [Hoeller Jr. and Fröhlich 2011].

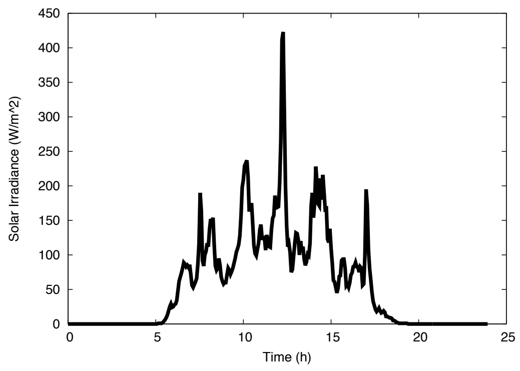
Energy production depends on the energy harvesting technology, and the used model should be detailed for each application. Regardless of the harvesting technology, however, the amount of energy available to the system in a moment ($E^i_{batt}$) can be estimated as shown in Equation 4. Given a previously known battery charge ($E^{(i-1)}_{batt}$), current battery charge comes from the subtraction of the amount of energy consumed in a period ($E^i_{tot}(\phi)$), and the sum of the amount of energy coming in from the harvesting system ($R_i$). As the amount of stored energy cannot be infinite, the computed battery charge has to be limited by the battery capacity ($E^{max}_{batt}$).

$$E^i_{batt} = \min\left(E^{max}_{batt}, E^{(i-1)}_{batt} - E^i_{tot}(\phi) + R_i\right) \tag{4}$$

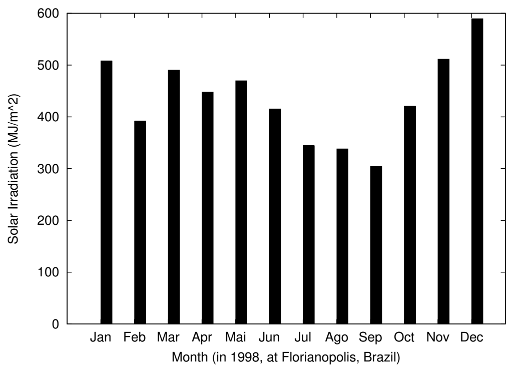
For example, energy production through photovoltaic modules, besides being subject to characteristics of the mechanisms themselves, is also dependent on solar irradiance and temperature. In solar systems, irradiance is the parameter that primarily influences energy production. Figure 1 shows that irradiance varies considerably during daytime (top), as does irradiation[1] during the year (bottom). It is possible to observe, however, that an important repetition pattern exist for sunlight irradiance: it is not present during nighttime and tends increase during the morning, until it reaches a peak around noon, to start to decrease again. In order to schedule a system in an adequate way, an energy-aware scheduler should be aware of such variations. By doing so, the system might store energy during the day to use at night, being, thus, able to reduce oscillations on system quality.

The size of the energy storage ($E^{max}_{batt}$ in Equation 4) is another key parameter in the present scenario once the system may not store all the excess energy due to the lack of capacity. There are other parameters affecting the performance of the energy storage in either rechargeable batteries or supercapacitors, such as capacity degradation, aging, and operating temperature. These issues, although relevant, are by now outside the scope of this work.

[1]Solar irradiation is the cumulative energy brought from solar irradiance over a period of time, i.e. $E_I = \int_{t_0}^{t_1} I dt$.



Solar irradiance of January 1st, 1998.



1998 monthly irradiation.

Fig. 1: Sample solar irradiance and irradiation at Florianópolis, Brazil ($27^o$ latitude).

## 3. SYSTEM DESIGN AND IMPLEMENTATION

The EPOS Project (Embedded Parallel Operating System) aims at automating the development of embedded systems so that developers can concentrate on the applications [LISHA 2012]. EPOS relies on the Application-Driven Embedded System Design (ADESD) method [Fröhlich 2001] to guide the development of both software and hardware components that are adaptable. The adaptation of components takes place during design and aims at fulfilling the requirements of applications. EPOS has a set of tools to support developers in selecting, configuring, and plugging components into an application-specific framework [Schulter et al. 2007]. The combination of methodology, components, frameworks, and tools enable the automatic generation of application-specific embedded system instances.

Besides the run time support system and tools, the EPOS Project has driven the development of hardware platforms, being the EPOSMOTE among them. EPOSMOTE is a modular platform for wireless sensor network applications. The implementation of the approach described in this paper used the EPOSMOTE platform. The implementation also modified EPOS's scheduler and power manager. The following sections describe the implementation.

### 3.1. EPOSMOTE

The EPOSMOTE is a modular platform for building wireless sensor network applications. Figure 2 shows the block diagram with the three modules of the platform. The `Processing Module` incorporates the core processing and communication components of the system. There are two different versions of this module: one based on Atmel ZigBit System-in-a-Package (SiP) and another, used in this work, based on Freescale MC13224V System-on-a-Chip (SoC). Both present RF transceivers compatible with IEEE 802.15.4 standard [IEEE 2006] and an integrated processor, which is an 8-bit AVR for the Atmel SiP and a 32-bit ARM7 for the Freescale SoC.

The application designer can adapt the hardware to the needs of each application through the power supply and IO interfaces factored out on EPOSMOTE design. The power interface features separate signals for the power source (i.e. $V_{cc}$, $V_{dd}$ and $Gnd$) and an $I^2C$ interface for communicating with the processing module. The IO interface has 34 pins available for custom designs, including a bypass of the power source, all $ADC$ channels, $SPI$, $UART$ and several $GPIO$ pins. The EPOSMOTE Project developed a *Start-Up* board to be connected to the IO interface that features a $USB$ converter, a

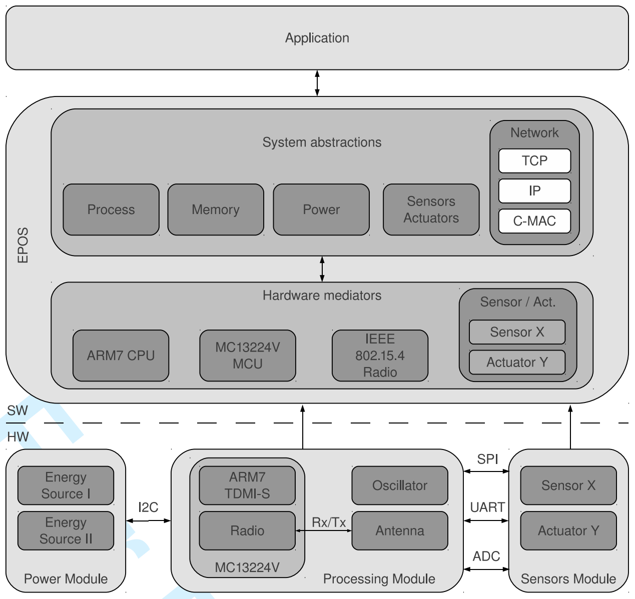


Fig. 2: Block diagram of the EPOSMOTE platform.

Table I: Current drain of CPU and Radio components of the EPOSMOTE.

| **Device/Mode** | **Current** ($\mu A$) |
|---|---|
| *CPU* | |
| Active | $2,200$ |
| Sleep | $40$ |
| Power Down | $3.4$ |
| *Radio* | |
| Transmit only | $19,300$ |
| Receive only | $16$ |
| Low Power Listen | $0.8$ |

Table II: Energy consumption of monitored events of the EPOSMOTE.

| **Event** | **Energy** ($\eta Ah$) |
|---|---|
| Read temperature sensor | 10.584 |
| Sample battery voltage | 0.462 |

thermistor, a 3-axis accelerometer, LEDs, and push buttons. This work also used the *Start-up* board.

To account for energy consumption of EPOSMOTE we first need to produce its power characterization and map it to the energy model described in Section 2.2. This information comes either from the components datasheets, when available, or from measurements in a real system. Table I shows values of current drains of system devices in different operating modes to be used by time-based accounters. Table II shows how much energy each monitored event consumes. The event accounting system uses this information to estimate the amount of consumed energy.

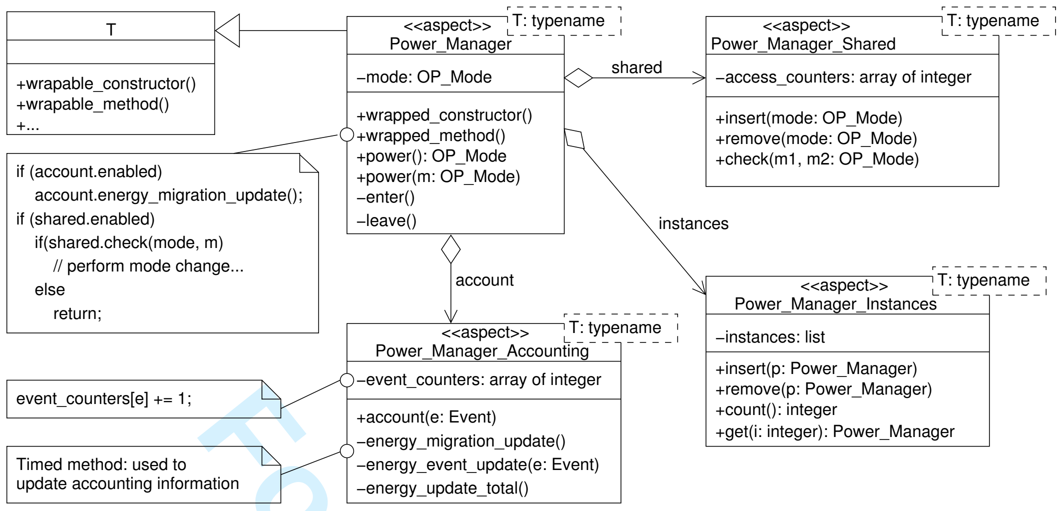


Fig. 3: UML class diagram of EPOS power manager.

### 3.2. EPOS Power Manager

Once energy is a non-functional property of computing systems [Lohmann et al. 2005], the design of EPOS power manager uses aspect-oriented programming [Mens et al. 1998] to implement its functionalities in a way that is orthogonal to other components of the operating system. EPOS implements aspects as constructs called *Scenario Adapters* [Fröhlich and Schröder-Preikschat 2000] that rely on the static metaprogramming capability of C++ (templates).

Figure 3 shows a class diagram of the EPOS power manager designed as a scenario adapter. The base class `Power_Manager` wraps the target class, i.e. the class that the aspect modifies. The wrapping happens through inheritance and function overriding (from the *Adapter Design Pattern* [Gamma et al. 1994]). Additional methods may be easily included, as is the case of the `power` methods in of the base `Power_Manager` class. `Power_Manager` is, in turn, a facade (from the *Facade Design Pattern* [Gamma et al. 1994]) to other power management functionalities implemented by other components. For instance, `Power_Manager_Shared` and `Power_Manager_Instances` are responsible for, respectively, controlling of operating modes for shared components, and keeping of object references for system-wide power management actions [Hoeller Jr. et al. 2006].

An extension to EPOS power manager, the `Power_Manager_Accounter`, implements the energy model described in Section 2.2. This extension enables the energy consumption accounting functionality in EPOS. The `account(e:Event)` method accounts the events using the event-based profile (Equation 2), which may be called in a wrapped method if the event generated by such a method is a monitored one. As concerning overhead issues, it is important to note that `account(e:Event)`, which increments an event counter, is an inline function, thus incurring in no overheads due to function calls at runtime. Also, the branches that implement the facade at `power(m:OP_Mode)` of `Power_Manager` use constant boolean values which EPOS defines at configuration time, before system generation. As such, these are subject to compiler optimizations that remove the branches in the final binaries of the system. Table III presents the impact of the proposed accounter in terms of code size and data memory usage. As can be seen, the accounting mechanism aggregates 2,768 bytes of code (ROM) and 70 bytes of data (RAM) to the original, fully functional `Power_Manager`.

Table III: Memory footprint of EPOS power manager.

| Setup | Sizes in bytes | | | |
|---|---|---|---|---|
| Section | .text | .data | .bss | **Total** |
| Original | 29,146 | 396 | 325 | **29,867** |
| with accounting | 31,914 | 454 | 337 | **32,705** |
| Increase | 9.5% | 9.71% | | **9.5%** |

### 3.3. EPOS Real-Time Scheduling

Figure 4 shows the three main components forming the real-time scheduling support on EPOS: `Thread`, `Criterion`, and `Scheduler`. The `Thread` class represents an aperiodic task and defines its execution flow, with its own context and stack. This class implements traditional thread functionalities, such as suspend, resume, sleep, and wake up operations. The `Periodic_Thread`[2] provides support for periodic tasks by extending the `Thread` class and aggregating mechanisms related to the re-execution of the periodic task. The `wait_next` method performs a `p` operation on a semaphore, forcing the thread to sleep until it reaches the next activation instant. Each periodic thread aggregates an `Alarm` object that is responsible for performing a `v` operation that, periodically, releases and wakes up the thread.

The `Scheduler` class and `Criterion` subclasses define the structure that realizes task scheduling. Usually, object-oriented OS scheduler implementations use a hierarchy of specialized classes of an abstract scheduler class. In this case, subclasses specialize the abstract class to provide different scheduling policies [Marcondes et al. 2009]. EPOS reduces the complexity of maintaining such hierarchy and promotes code reuse by detaching the scheduling policy (here represented by the `Criterion` subclasses) from its mechanism (e.g., data structure implementations as lists and heaps). The data structure in the scheduler class uses the defined scheduling criterion to order the tasks accordingly. During compilation, the `Trait`[3] class of `Thread` defines the scheduling criterion. For example, `typedef Scheduling_Criteria::EAEDF Criterion` defines the scheduling criterion as Energy-Aware EDF. The `Scheduler` consults the information that the criterion class provides to define the appropriate use of lists and operations.

Each criterion class defines the priority of a task, which the scheduler uses to choose a task (`operator ()`), and other criterion features, such as preemption and timing, for instance. In this work, we extended the EDF and RM criterions to support energy-aware operation. We created a flag (`ENERGY_AWARE`) that informs the scheduler whether the criterion is energy-aware or not. As shown in the sequence diagram of Figure 5, the scheduler uses the flag to decide whether it must check for energy availability or not before dispatching a best-effort task. It is important to highlight that flags are static and constant values[4] and, due to this reason, the compiler optimizes the if-statements that verify whether the criterion in use is energy-aware or not, reducing the runtime overhead.

With this separation of concerns among scheduler, criterion, and thread, it is straightforward to add new scheduling policies into the system. Moreover, in cases where a scheduling policy requires specific scheduling treatment, a new scheduler may be created by extending the existing schedulers through metaprogramming specialization techniques [Czarnecki and Eisenecker 2000].

[2]A periodic thread in EPOS is conceptually equivalent to a real-time periodic task.
[3]A trait class is a template class that associates information of a component at compile time.
[4]In C++: `static const bool ENERGY_AWARE = true;`

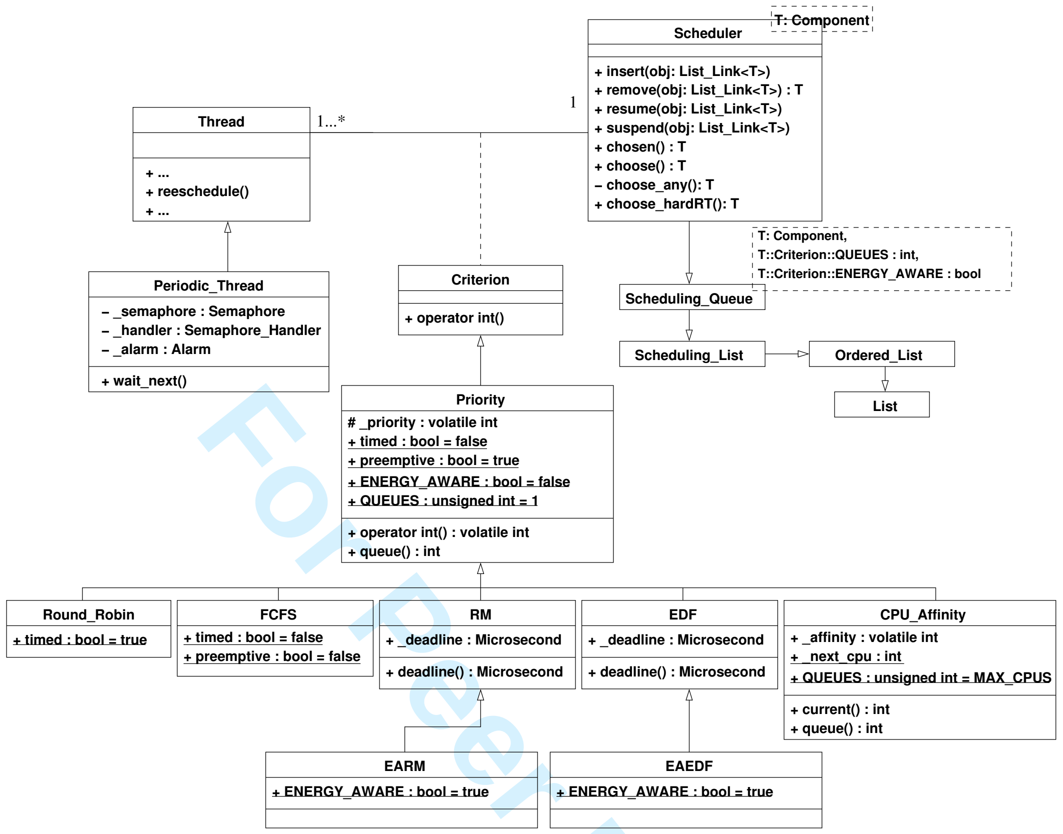

Scheduler
T: Component
+ insert(obj: List_Link<T>)
+ remove(obj: List_Link<T>) : T
+ resume(obj: List_Link<T>)
+ suspend(obj: List_Link<T>)
+ chosen() : T
+ choose() : T
– choose_any(): T
+ choose_hardRT(): T
Thread
+ ...
+ reeschedule()
+ ...
1...*
1
Periodic_Thread
– _semaphore : Semaphore
– _handler : Semaphore_Handler
– _alarm : Alarm
+ wait_next()
Criterion
+ operator int()
T: Component,
T::Criterion::QUEUES : int,
T::Criterion::ENERGY_AWARE : bool
Scheduling_Queue
Scheduling_List
Ordered_List
List
Priority
# _priority : volatile int
+ timed : bool = false
+ preemptive : bool = true
+ ENERGY_AWARE : bool = false
+ QUEUES : unsigned int = 1
+ operator int() : volatile int
+ queue() : int
Round_Robin
+ timed : bool = true
FCFS
+ timed : bool = false
+ preemptive : bool = false
RM
+ _deadline : Microsecond
+ deadline() : Microsecond
EDF
+ _deadline : Microsecond
+ deadline() : Microsecond
CPU_Affinity
+ _affinity : volatile int
+ _next_cpu : int
+ QUEUES : unsigned int = MAX_CPUS
+ current() : int
+ queue() : int
EARM
+ ENERGY_AWARE : bool = true
EAEDF
+ ENERGY_AWARE : bool = true

Fig. 4: UML class diagram of the scheduling structure of EPOS.

## 4. ADAPTIVELY SUSTAINING SYSTEM OPERATION

Systems featuring rechargeable energy sources have their lifetime bounded to their ability to sustain the energy needed for its operation. This contrasts, in part, to the traditional concept that the lifetime of a wireless sensor network without renewable energy sources is bounded to the lifetime of the batteries of its nodes. In the context of real-time systems, the successful operation of a system depends on the compliance to the requirements of its critical parts. As a consequence, an energy-aware scheduler for real-time systems must somehow guarantee that there will be enough energy to supply the critical parts of the system.

A basic approach for guaranteeing battery lifetime in a system is defining the maximum discharge rate of subsequent iterations ($D_{i+1}$). This rate is a function of current battery charge ($E^i_{batt}$) and remaining required operation time ($t_r$ is the required time), as shown in Equation 5. An energy overload takes place when the system demand for energy surpasses this limit. In this situation, it becomes impossible to keep the energy consumption bellow $D_i$ without, somehow, adapting the system. Here, a scheduler adjusts periodically the execution rate of best-effort tasks (called “BET rate” from now on) to the maximum rate satisfying the computed battery discharge rate.

$$D_{i+1} = \frac{E^i_{batt}}{t_r - t_i} \quad (5)$$

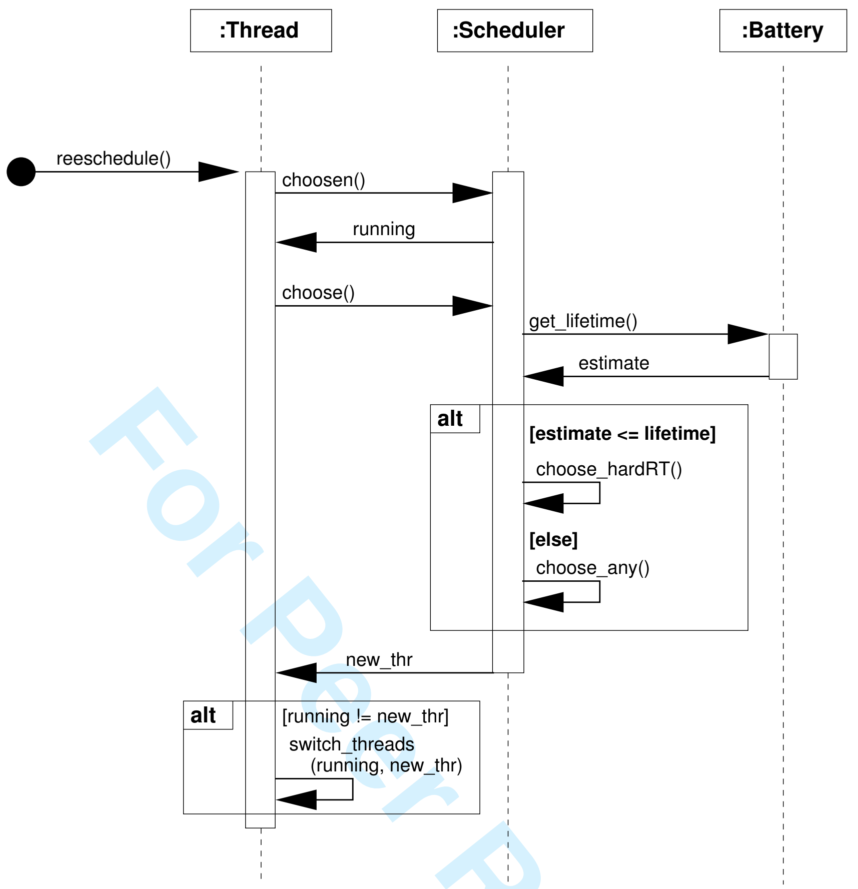


Fig. 5: UML sequence diagram of the adaptation of the scheduling mechanism of EPOS.

One assumption of the proposed energy-aware scheduler is that 100% of the hard real-time tasks execute until the system reaches the targeted lifetime. Consequently, the scheduler needs to reserve energy for those tasks, what makes all scheduling approaches studied in this work conservative. This reserve ($E_{res}$), as Equation 6[5] shows, is the sum of the worst-case energy consumption ($E_{\Gamma_j}$) of all instances of hard real-time tasks ($\Gamma$) to be released until the system reaches the targeted lifetime ($t_r$).

$$E^i_{res} = \sum_{j=0}^{|\Gamma|} \frac{t_r - t_i}{T_{\Gamma_j}} \cdot E_{\Gamma_j} \tag{6}$$

The following paragraphs describe the four approaches evaluated in this work. The approaches follow the premises specified on Equations 5 and 6. This means that all proposed approaches will reserve energy for the hard real-time tasks and allocate excess energy to BET tasks.

Equation 7 defines the first approach: *BET Global Rate* (BGR). BGR allocates all excess energy (i.e. energy available for BET tasks) to the BET tasks that will run in

[5] $\Gamma$ is the set of hard real-time tasks; $T_{\Gamma_j}$ is the period of the $jth$ task of $\Gamma$; and $E_{\Gamma_j}$ is the worst-case energy consumption of the $jth$ task of $\Gamma$.

the next iteration. At first sight, it might not be a wise decision once the system may run out of excess energy very quickly. It may not happen, however, in systems where incoming energy exceed consumed energy in an iteration period. Another drawback of this approach may be its deployment in systems where energy income is sporadic or periodic. For instance, a photovoltaic system using this approach may perform well during the day but may run out of excess energy during the night due to the absence of incoming energy. The same may happen with wind turbine systems when a long lull period happens.

$$BGR_i = \min\left(1.0\ ,\ \frac{E^i_{batt} - E^i_{res}}{\sum_{j=0}^{|\beta|} \frac{(t_{(i+1)} - t_i)}{T_{\beta_j}} \cdot E_{\beta_j}}\right) \tag{7}$$

Equation 8 presents an approach to prevent the system from running out of excess energy during periods of low energy income and, also, to make a fairer distribution of BET rate along time. This heuristic, the *BET Window Rate* (BWR), divides the excess energy among distinct time windows ($w_s$). Although "fairer", this approach may negatively affect the BET rate depending on the amount of excess energy available to the system, i.e. the size of the energy storage. It is also sensitive to the relation between the desired lifetime ($t_r$) and time window size ($w_s$).

$$BWR_i = \min\left(1.0\ ,\ \frac{\frac{E^i_{batt} - E^i_{res}}{(t_r - t_i) \div w_s}}{\sum_{j=0}^{|\beta|} \frac{(t_{(i+1)} - t_i)}{T_{\beta_j}} \cdot E_{\beta_j}}\right) \tag{8}$$

Two other approaches address the enhancement of the performance of the energy allocator to handle sporadic and periodic energy sources. Both approaches deal with variations in energy income spreading the excess energy evenly across the remaining operation time of the system. This may prevent the system from passing long periods of time without being able to execute BET tasks. Another addition to these policies is that both approaches try to predict the short-time future of energy income based on recent history. This is particularly useful for systems where the storage (i.e. battery) cannot accommodate all the incoming energy, because it helps to reduce the wasted energy.

Equation 9 shows the *BET Window Average Rate* (BWAR). BWAR tries to raise BET rate by predicting that the energy income for the next iteration will be the average of the energy income of the previous iterations. In the equation, $R$ is the series of incoming energy, $w$ is the number of previous iterations taken into account for the average, and $w_s$ is the length of each iteration. This approach may perform well for sporadic energy sources and aleatory energy sources, such as wind. For periodic energy sources such as the solar-based ones, the average may render several errors. Moreover, the prediction accuracy of this method is highly dependent on the size of the look-back window. As can be seen in Figure 1 (top), although interferences exist (e.g. clouds blocking sunlight), the tendency of the solar irradiance during a day shows a forth-order shape, raising during the first half of the day and lowering on the second half. Thus, using recent average may be conservative during the first half of the day, and too aggressive during the second half.

$$BWAR_i = \min\left(1.0\ ,\ \frac{\frac{E^i_{batt} - E^i_{res}}{(t_r - t_i) \div w_s} + \frac{\sum_{j=i-(w+1)}^{i-1} R_j}{w}}{\sum_{j=0}^{|\beta|} \frac{(t_{(i+1)} - t_i)}{T_{\beta_j}} \cdot E_{\beta_j}}\right) \tag{9}$$

To cope with the cited limitations of BWAR, *BET Window Derivative Rate* (BWDR), shown in Equation 10, tries to reduce the prediction errors by taking the energy income derivative into account instead of the recent average. In the equation, $R_{i-1}$, which is the amount of income energy in the last period, is summed to the expected variation in the energy income for the current period, i.e. its derivative. In this heuristic, the Newton's difference quotient method (the last fraction), which is a straightforward two-point estimation to compute the slope of a nearby secant line through the previous two points of the energy input curve, numerically approaches this derivative.

$$BWDR_i = \min\left( 1.0\ ,\ \frac{\frac{E^i_{batt} - E^i_{res}}{(t_r - t_i) \div w_s} + R_{i-1} + \frac{R_{i-1} - R_{i-2}}{t_{i-1} - t_{i-2}}}{\sum_{j=0}^{|\beta|} \frac{(t_{(i+1)} - t_i)}{T_{\beta_j}} \cdot E_{\beta_j}} \right) \tag{10}$$

## 5. CASE STUDY: SCHEDULING IN A MOBILE WSN

The power management mechanism described in Section 3.2 incorporates the proposed heuristics. Although having a real implementation of the system, the evaluation of the heuristics uses simulation. This is because of the nature of the studied system, which includes yearlong analysis of system behavior to verify the efficiency of the energy schedulers. The simulation presented in this paper is that of a dense, mobile network. This application builds a critical scenario for communication and energy consumption due to the large amount of data exchanged in the network. The simulation parameters receive data collected from real measurements performed on EPOSMOTE nodes powered by rechargeable batteries and solar panels.

The application is a WSN running the Ant-based Dynamic Hop Optimization Protocol (ADHOP) [Okazaki and Fröhlich 2011] over an IP network using IEEE 802.15.4. ADHOP is a self-configuring, reactive routing protocol. The reactive component of ADHOP uses an *Ant Colony Optimization* algorithm to discover and maintain routes. Ants go out to track routes, leaving a trail of pheromone on their way back. The selected routes for data exchange are the ones with a higher pheromone deposit.

With the purpose of evaluating the energy scheduling heuristics presented in this paper, we modified ADHOP in order to make it energy-aware. The tasks implementing ADHOP are now either hard real-time or best-effort, as described in the application model (Section 2.1). The main idea behind this setup is to homogenize the battery discharge for every node in the network to enhance the lifetime of the network as a whole. Considering the radio as the most energy-hungry component in a wireless sensing node, we toke the design decision of modeling the routing activity of ADHOP as a best-effort task, as shown by the task set at Table IV. The basic node functionalities of sensing a value (task $Sense$) and sending it through the radio to a sink node (task $Send$) where modeled as hard real-time tasks. Two best-effort tasks implement the functionality of forwarding packets (and ants) coming from other nodes when acting as a "router": one monitors the channel for arriving messages ($LPL$ - Low Power Listen) and another receives messages and route them to another node ($Route$). This approach will cause links to go down when a node realizes it must stop best-effort tasks. However, ADHOP is a suitable candidate for these modifications because the ACO routing mechanism tries to find alternative routes if a link fails.

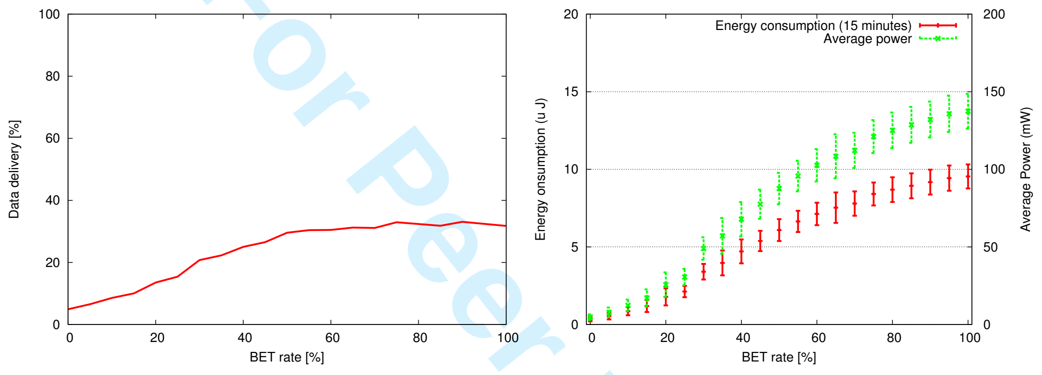
We fixed the lifetime objective for this system at 30 days. This means that, all along the system execution, if the energy harvesting mechanism fails, the system will still be able to run its hard real-time tasks for, at least, 30 days. It is possible to analyze the task set and compute the total energy consumption of the hard real-time tasks to be around $722\ mAh$ during the desired lifetime. Thus, the initial battery charge for the system has to be greater than that to allow the system to reserve energy for the critical part of the application.

The evaluation procedure comprises two simulation steps. In the first step, a simulation using the OMNET++ Simulator characterized response of the application to

Table IV: Parameters for the tasks of the ADHOP case-study.

| **T** | **P** ($ms$) | **WCET** ($ms$) | **WCEC** ($\eta Ah$) | **30-days** ($mAh$) |
|---|---|---|---|---|
| $Sense$ | $1,000$ | 2 | 10.58 | 27.43 |
| $Send$ | $1,000$ | 50 | 268.06 | 694.8 |
| $Collector$ | 5 | $45.3 \times 10^{-6}$ | $4.16 \times 10^{-6}$ | $2.15 \times 10^{-3}$ |
| $LPL$ | 5 | 0.25 | $1.11 \times 10^{-6}$ | $0.58 \times 10^{-3}$ |
| $Route$ | 50 | 100 | 490.278 | $25.42 \times 10^{+3}$ |
| **Energy consumption of hard real-time tasks** | | | | 722.24 |

*Legend:* T: task; P: period in $ms$; WCET: worst-case execution time in $ms$; WCEC: worst-case energy consumption in $\eta Ah$; 25-days: energy consumption for the targeted lifetime (25 days) in $mAh$.
*Route:* The "Route" task is a sporadic task. Once it is a best-effort task, we consider a hypothetic frequency of 2 Hz (period of 500 $ms$) to show the impact of routing in energy consumption.



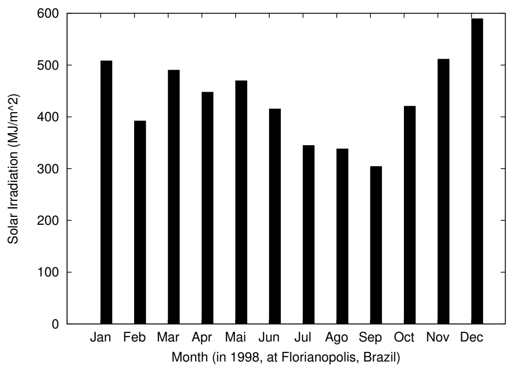
Data delivery per BET rate. Avg. power and energy cons. per BET rate

Fig. 6: Network response to BET rate.

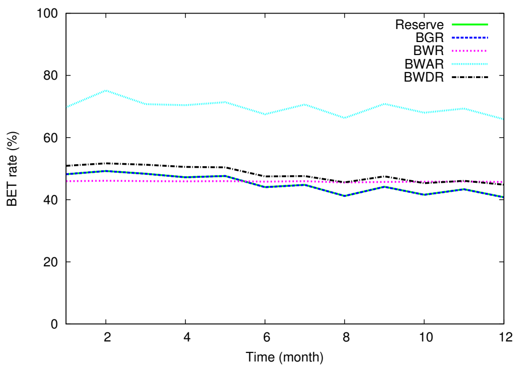
variations in the execution rate of best-effort tasks (BET rate). As can be seen in Figure 6, lower energy consumption at lower BET rates comes at the cost of lower data delivery rate.

The second step applies the proposed heuristics to analyze the energy consumption over one year. Besides the proposed heuristics, the simulations cover two other situations. In the first one, the scheduler does not impose any constraint on the BET rate. Although this showed a large gain in the BET rate, it also caused real-time tasks to fail due to the lack of energy. It shows that the system needs to control energy allocation to handle hard real-time applications. The second one is a reserve-based approach explored in previous experiments [Hoeller Jr. and Fröhlich 2011]. In this approach, BET rate is either $100\%$ or $0\%$, depending on the system energy status. Although this second approach respects the deadline of hard real-time tasks, its wide variation range brings an undesirable instability to the BET rate.

Each one of the heuristics proposed here had its parameters set to the best scenario for its execution. For BGR, the system uses a smaller battery once it shows no large dependency on energy buffer size. For BWR, however, the system needs a large battery buffer to assure a reasonable BET rate. That would be a problem if space or weight become an issue for this system as, for instance, 12 AA-size batteries would be needed to provide a system with 12,600 mAh at 3 V. BWAR and BWDR, although also dependent on the energy buffer size, do not demand extremely large buffers to exist once, with



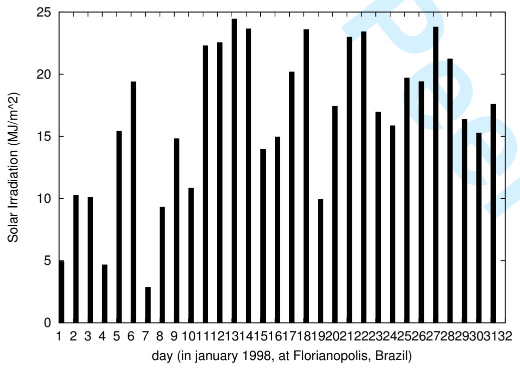
1998 monthly irradiation.



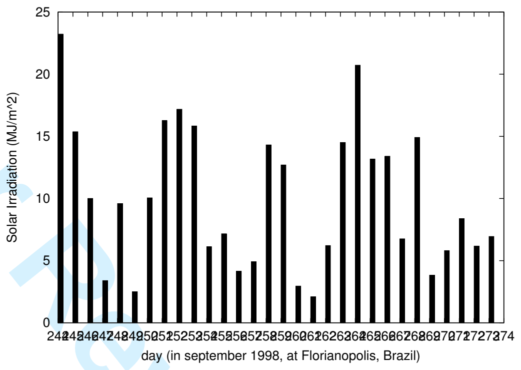
BET rate during 1998.

Fig. 7: BET rate variations over the 1-year simulation.

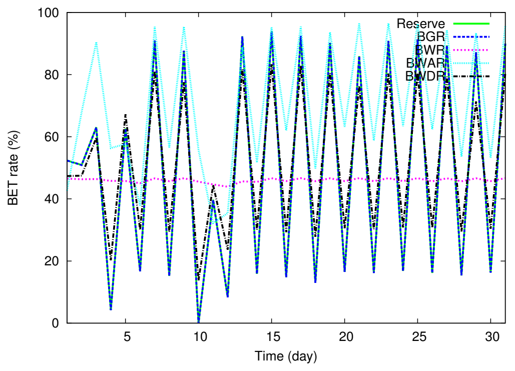
the assumption of incoming energy from the energy-harvesting mechanism, they are able to compensate for the low share of stored energy they use.



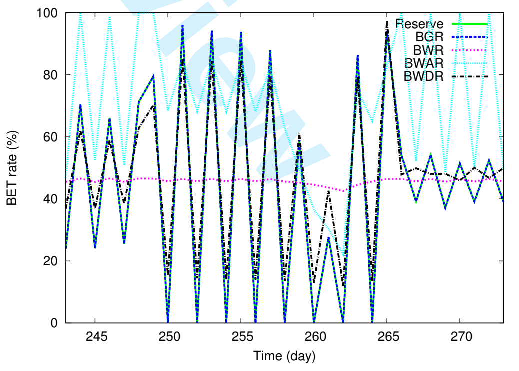
January 1998 daily irradiation.



September 1998 daily irradiation.

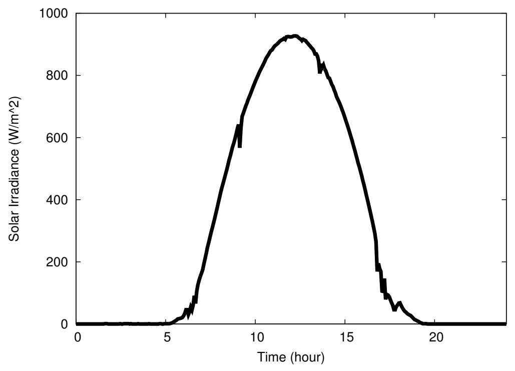


BET rate during January 1998.

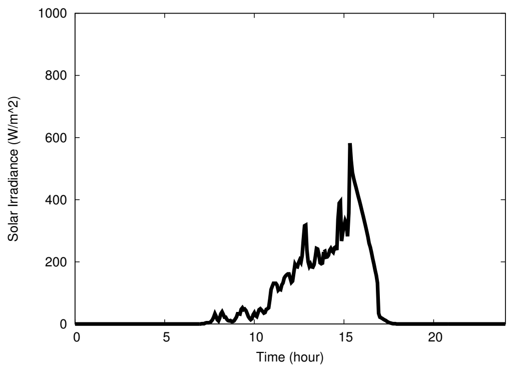


BET rate during September 1998.

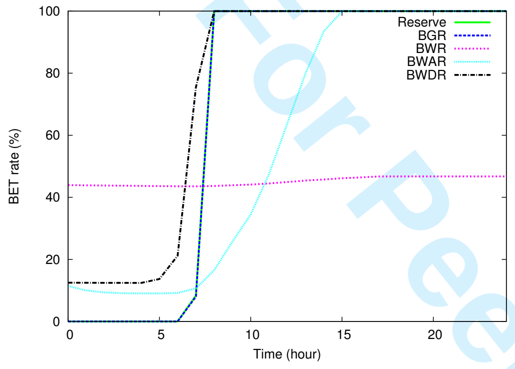
Fig. 8: BET rate variations over a summer and a winter months.



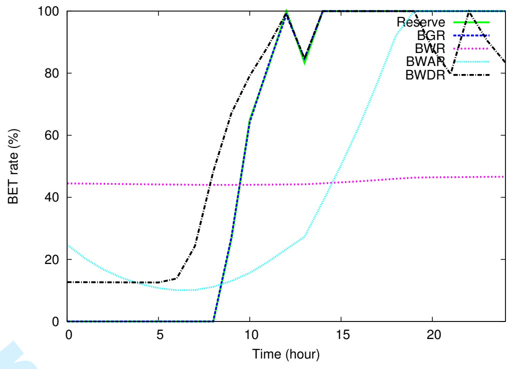
Solar irradiance during summer days.



Solar irradiance during winter days.



BET rate during summer days.



BET rate during winter days.

Fig. 9: BET rate variations around the southern (summer) and northern (winter) solstices in the southern hemisphere.

In the simulations, the hard real-time tasks fail often when the system does not perform energy scheduling. This happens because the system runs out of energy during periods of low solar irradiation or at night. It is also necessary to note that, on the event of a longer failure on the energy harvesting mechanism, the system would go totally down in few hours if it does not reserve energy.

Figure 7 shows the variations on the monthly BET rate over one year. As can be seen, regardless of how the approaches try to reduce the impact of the lower solar irradiation in the second half of the year, all of them slightly decrease BET rate due to the diminished amount of energy coming into the system. That variation had only been small, however, if taking into account the monthly BET rate. Figure 8 shows the variation of the BET rate during a summer and winter months. As can be seen, system response to irradiation is fast in all approaches except BWR. That happens because BWR does not speculate over energy production and features a larger energy buffer. It is also possible to note that BWAR, although presenting an operating amplitude of around 60 p.p., shows a better performance, keeping BET rate above BWR most of the time without the need for a larger energy buffer.

Figure 9 shows another characteristic of the simulated schedulers. These figures show the BET rate around the southern (December 21st) the northern (June 20nd) solstices, which in southern Brazil should be periods of, respectively, higher and lower solar irradiation levels. As can be seen, all approaches, except BWR and BWAR, nulled BET rate on a daily basis.

Summarizing the results, *BET Window Rate* (BWR) and *BET Window Average Rate* (BWAR) behave in much better way when compared with others. BWR shows a remarkably stable behavior at the cost of higher demand of battery size. BWAR, although showing a large operating amplitude, was able to execute more best-effort tasks than the others. Also, during the whole year, the BET rate obtained through BWAR was never zero.

## 6. RELATED WORK

Several works have addressed the deployment of energy harvesters in wireless sensor networks. Although many of them did not take real-time constraints into account, some can be adapted to support timing requirements. This section describes the approaches these works used in order to schedule energy usage in energy-harvesting systems.

An important factor for efficiently managing energy in energy-harvesting systems is the choice of an efficient predictor for energy input [Paradiso and Starner 2005]. Kansal et al. [Kansal et al. 2007] introduced a prediction algorithm to support their power management approach for wireless sensor networks. Their energy prediction model uses an exponentially weighted moving-average filter. This method assumes that solar irradiation in a given time-slot of the day is similar to the irradiation in the same time-slot of other days. Their power management approach took into account the predicted amount of energy input to maximize the duty-cycle of the system, subject to energy availability. Moser et al. [Moser et al. 2007] used a similar approach. Their work presents a system model, simulated experimental results, and a real implementation of the proposed control law. The shown experiments, however, focus on a single sensor node, not taking wireless communication into account. Ali et al. [Ali et al. 2010] proposed an extension to these models. Their approach take not only the past days into account, but also the solar irradiation levels of the current day. This allows for lower prediction errors as, by monitoring the irradiation levels of the current day, it is possible to adjust predicted energy according to the most recent information, i.e., the prediction errors in the most recent time-slots.

A few works analyzed the problem of efficiently scheduling tasks in systems presenting varying energy budgets. Rusu et al. [Rusu et al. 2003] presents a multiversion scheduling approach that chooses between the different versions of the same tasks that maximize the system value and prevents battery depletion. Assuming that batteries may be recharged, they propose a static solution that maximizes value based on worst-case scenarios. Also, they propose a dynamic scheme that takes advantage of the slack energy generated in the system when worst-case energy consumption does not take place. El Ghor et al. [EL Ghor et al. 2011] proposed a modified EDF scheduler that takes energy production, storage, and consumption into account. They show by simulation that the approach outperforms other algorithms in terms of rate of deadline misses and required size of energy storage. Sharma et al. [Sharma et al. 2010] analyze the problem from a different perspective. They consider that the function of the device in a WSN is solely to sense and to forward data through the wireless interface. In this context, they propose a set of policies to minimize the size of an outgoing queue subject to energy availability. They assume that, the smaller the queue size, the larger the amount of data forwarded through the network, and, consequently, the better the system quality.

From all studied related work, only Kansal et al. [Kansal et al. 2007] have analyzed the impact of their work in terms network-wide performance. Other works only analyze the performance of isolated energy-harvesting sensor nodes.

## 7. CONCLUSION

This paper presented the evaluation of four heuristic approaches for energy allocation in real-time wireless sensor network systems that harvest energy from the environment. A specific task model is assumed in which system tasks may be classified as critical or noncritical. Critical parts are hard real-time tasks and must be assured energy and processing time for execution. Noncritical parts are best-effort tasks and have they execution granted when there is enough energy for doing so. The four heuristics were evaluated in a realistic simulation environment considering a dense, mobile, wireless sensor network application. Simulations show that all heuristics respect the real-time constraints of the system. The efficiency metric used for evaluation was the rate of best-effort tasks executed by the system.

Results show that two of the heuristics, *BET Window Rate* (BWR) and *BET Window Average Rate* (BWAR), behave in much better way when compared with others. BWR shows a highly stable behavior at the cost of higher demand of battery size. BWAR, although showing a large operating amplitude, was able to execute more best-effort tasks than the others. Also, during the whole year, the rate calculated by BWAR was never zero.

Ongoing work is evaluating the same approaches using wind turbines instead of solar panels as the energy source. It is expected that the behavior of the schedulers will be considerably distinct from the ones shown here because of the chaotic behavior of wind speed.

## REFERENCES

ALI, M. I., AL-HASHIMI, B. M., RECAS, J., AND ATIENZA, D. 2010. Evaluation and design exploration of solar harvested-energy prediction algorithm. In *Proceedings of the Conference on Design, Automation and Test in Europe*. DATE '10. European Design and Automation Association, 3001 Leuven, Belgium, Belgium, 142–147.

CZARNECKI, K. AND EISENECKER, U. W. 2000. *Generative programming: methods, tools, and applications*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

EL GHOR, H., CHETTO, M., AND CHEHADE, R. H. 2011. A real-time scheduling framework for embedded systems with environmental energy harvesting. *Comput. Electr. Eng. 37,* 4, 498–510.

FAY, D. AND RINGWOOD, J. 2010. On the influence of weather forecast errors in short-term load forecasting models. *Power Systems, IEEE Transactions on 25,* 3, 1751 –1758.

FRÖHLICH, A. A. 2001. *Application-Oriented Operating Systems*. GMD - Forschungszentrum Informationstechnik, Sankt Augustin.

FRÖHLICH, A. A. 2011. A Comprehensive Approach to Power Management in Embedded Systems. *International Journal of Distributed Sensor Networks 2011,* 1, 19.

FRÖHLICH, A. A. AND SCHRÖDER-PREIKSCHAT, W. 2000. Scenario Adapters: Efficiently Adapting Components. In *4th World Multiconference on Systemics, Cybernetics and Informatics*. Orlando, USA.

GAMMA, E., HELM, R., JOHNSON, R., AND VLISSIDES, J. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, USA.

HOELLER JR., A. AND FRÖHLICH, A. A. 2011. On the Monitoring of System-Level Energy Consumption of Battery-Powered Embedded Systems. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*. Anchorage, AK, USA, 2608–2613.

HOELLER JR., A., WANNER, L. F., AND FRÖHLICH, A. A. 2006. A Hierarchical Approach For Power Management on Mobile Embedded Systems. In *5th IFIP Working Conference on Distributed and Parallel Embedded Systems*. Braga, Portugal, 265–274.

IEEE. 2006. Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low rate wireless personal area networks (wpans). *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, 1 –320.

KANSAL, A., HSU, J., ZAHEDI, S., AND SRIVASTAVA, M. B. 2007. Power management in energy harvesting sensor networks. *ACM Trans. Embed. Comput. Syst. 6,* 4.

LISHA. 2012. EPOS project website. Internet. http://epos.lisha.ufsc.br.

LIU, C. L. AND LAYLAND, J. W. 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM 20,* 1, 46–61.

LOHMANN, D., SCHRODER-PREIKSCHAT, W., AND SPINCZYK, O. 2005. 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*. 413–420.

MARCONDES, H., CANCIAN, R., STEMMER, M., AND FRÖHLICH, A. A. 2009. On design of flexible real time schedulers for embedded systems. In *International Symposium on Embedded and Pervasive Systems*. Vancouver, Canada, 382–387.

MENS, K., LOPES, C. V., TEKINERDOGAN, B., AND KICZALES, G. 1998. Aspect-oriented programming workshop report. In *Proceedings of the Workshops on Object-Oriented Technology*. ECOOP '97. Springer-Verlag, London, UK, 483–496.

MOSER, C., THIELE, L., BRUNELLI, D., AND BENINI, L. 2007. Adaptive power management in energy harvesting systems. In *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*. 1 –6.

OKAZAKI, A. M. AND FRÖHLICH, A. A. 2011. Ant-based dynamic hop optimization protocol: a routing algorithm for mobile wireless sensor networks. In *Joint Workshop of SCPA 2011 and SaCoNAS 2011 - IEEE GLOBECOM 2011*. Huston, Texas, USA, 1179–1183.

PARADISO, J. AND STARNER, T. 2005. Energy scavenging for mobile and wireless electronics. *Pervasive Computing, IEEE 4,* 1, 18 – 27.

PILLAI, P. AND SHIN, K. G. 2001. Real-time dynamic voltage scaling for low-power embedded operating systems. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*. ACM Press, New York, NY, USA, 89–102.

RUSU, C., MELHEM, R., AND MOSSE, D. 2003. Multiversion scheduling in rechargeable energy-aware real-time systems. In *Real-Time Systems, 2003. Proceedings. 15th Euromicro Conference on*. 95 – 104.

SCHULTER, A., CANCIAN, R., STEMMER, M. R., AND FRÖHLICH, A. A. M. 2007. A Tool for Supporting and Automating the Development of Component-based Embedded Systems. *Journal of Object Technology 6,* 9, 399–416.

SHARMA, V., MUKHERJI, U., JOSEPH, V., AND GUPTA, S. 2010. Optimal energy management policies for energy harvesting sensor nodes. *Wireless Communications, IEEE Transactions on 9,* 4, 1326 –1336.

Received October 2012; revised ; accepted