A Scheduling Algorithm for Consistent Monitoring Results with Solar Powered High-Performance Wireless Embedded Systems

Denis Dondi, Piero Zappi, Tajana Šimunić Rosing
Computer Science and Engineering (CSE)
University of California San Diego (UCSD)
92093 San Diego, CA
dondidenis@gmail.com, {pzappi, tajana}@ucsd.edu

***Abstract*— Getting consistent results when monitoring phenomena is a challenging but critical task for solar powered wireless high power embedded systems. Our algorithm relies on an energy predictor to achieve uniform monitoring over time while maximizing the number of tasks executed. Our approach outperforms state of the art algorithms by increasing the number of daily measurement by 30% and reducing their standard deviation by 5.5x.**

High Performance Wireless Embedded Systems, Task Scheduling Algorithm, Energy Harvesting and Energy Prediction

I. INTRODUCTION

High performances Wireless Embedded Systems are a natural choice for monitoring applications which generate and analyze large amounts of data. Such applications include monitoring of large structures [1], habitats [2] and for surveillance [18]. For such systems, there are typically twofold objectives: (1) energy consumption must be reduced to extend the battery lifetime, and (2) monitoring should be performed at a constant accuracy level over time. The latter objective is critical when we are interested in the evolution of a certain phenomenon or in detecting sporadic events that can happen at any time. Therefore, it is more beneficial to have a uniform quality of measurements and analysis over time rather than a burst of highly accurate data followed by period where the system does not have enough energy to provide any results. Some examples include: Structural Health Monitoring (SHM) in different environmental conditions (e.g. 24hr periods) and wireless video surveillance applications [19]. In the latter case, events of interest happen randomly, thus period of inactivity may result in critical event loss. Therefore, consistent monitoring is needed**.**

When large amount of data needs to be forwarded to a backend infrastructure for analysis through a multi-hop wireless sensor network, wireless communication becomes the major source of energy consumption [3]. Local computation is a common approach to reduce this contribution [4][5]. A number of monitoring applications require high computational capabilities to perform the desired data analysis. For example, ultrasonic based SHM relies on the analysis of ultrasonic waves propagated through the structure. Algorithms for the processing of such waveforms include frequency and time domain signal analysis [6] or video analysis [7]. For example, the sensor node for SHM presented [16] embeds 16 high voltage piezoelectric sensors that generate ultrasonic waves. These waves are sampled at high frequency (25 mega samples per seconds) resulting in 2 Mbytes of data every second. This volume of data cannot be locally processed by a low power, low capability Mote (see Table 1) and requires a powerful device such as a DSP running at 300MHz. Thus, for these kinds of systems, a number of high performance embedded components have been used including DSPs [8] and high-end microcontrollers [9]. These devices typically have Dynamic Voltage Frequency Scaling (DVFS) capabilities to improve the energy efficiency. However, since such devices remain in the environment, away from a stable power source, energy harvesting (EH) can be used to either provide its only source of energy or to extend the battery lifetime.

TABLE 1: COMPARISON OF THE CHARACTERISTIC OF A TYPICAL MOTE (BTNODE [22]) AND THE SHIMMER PLATFORM FOR SHM [16]

	BT Nodes	**Shimmer**
CPU	8 bit MCU	32 bit DSP
Data memory (kB)	128	4000
RAM (kB)	64+180	64000
Max Freq. (MHz)	8	300
DVFS capable	no	yes
Sensors	External	16 PZT sensors
Data generated	Few bytes per second	Up to 2MB per second
Peak power (mW)	198	2200

Solar harvesting is one of the most effective solutions for embedded systems placed outdoors [10]. A key challenge is that solar energy availability has non deterministic behavior [12] and the energy income of a photovoltaic module can be only estimated [11]. Several examples of solar powered Wireless Sensor Networks (WSN) have been presented in the literature [20]**[**21]. Still, these papers focus on solutions that are targeted for low power, low capabilities motes. Other works focus on adapting tasks execution of high performance embedded systems to the environmental energy availability [13][14]. However, this leads to a large variation on the quality of results obtained at different irradiance levels. The scheduling algorithm for energy harvesting device presented in [13] aims to execute application's tasks given a set of time and energy constraints. The scheduler relies on DVFS and is based on a static analysis of the worst case scenario. Three heuristics to allocate tasks have been defined and compared to distribute the extra energy obtained at a certain point in time over the following few time frames. In the proposed scenario non critical tasks can be dropped when energy is not available. This may lead to the case of some tasks never being executed. In [14] the task manager tries to schedule all the tasks at the lowest possible speed to save power using a lazy schedule. The task scheduler updates the system frequency and changes the voltage depending on the available energy, leading to high

978-1-61284-660-6/11/$26.00 © 2011 IEEE

variance in how much work gets done during any time frame. In all of previous work task allocation is performed based on current available energy (Greedy-like strategies), resulting in a task execution clustered around peaks of incoming energy. To the best of our knowledge, no solutions have been proposed to address the problem of consistent monitoring for high performance, solar powered wireless embedded systems (e.g. those used in high fidelity SHM monitoring or video surveillance).

The scheduler proposed in our paper specifically focuses on a set of applications whose aim is to keep the workload of solar powered wireless embedded systems consistent over time. It relies on a two stage process to allocate the task execution. The first stage includes the prediction of the daily energy availability from the solar harvester and the even allocation of energy resources (energy currently stored plus expected energy income) throughout the day. Therefore, even if the available energy at current time instance is low, when the predictor foresees that high energy income will be available in the future, the task manager increases the resources allocated at current time. The second stage optimizes the task execution according to the allocated energy and leverages system DVFS capabilities for higher energy efficiency.

To validate the proposed task scheduler, we simulated the performances of an embedded platform for active Structural Health Monitoring (SHM) [16]. This device generates and applies ultrasonic waves to the target structure, samples and analyzes locally the resulting, distorted waveform to detect the presence of faults. The node is powered through a solar EH combined with a supercapacitor that stores and distributes energy over time. Two applications are performed: periodic analysis to regularly monitor the status of the structure, and a high priority on-demand analysis that responds to catastrophic events. The accuracy of each analysis increases with the number of measurements performed (each measurement is considered an independent task). A utility function has been defined to evaluate the effectiveness of our approach. Our solution is compared to a Greedy solution (similar to [13]), to an energy-driven solution designed for harvester powered monitoring application [8], and to the oracle solution. In the oracle solution the scheduler knows exactly: 1) on-demand analysis arrival time, 2) the profile of the future incoming energy and 3) the evaluation function to be optimized. Simulation results show the effectiveness of the predictor-driven approach in terms of maximizing the utility function.

The rest of the paper is organized as follows. Section II describes the solar energy predictor used by the proposed task scheduling algorithm, which is presented in Section 0. In Section IV we show the experimental results obtained with our scheduler. Finally, Section V concludes the paper.

II. SOLAR ENERGY PREDICTOR

Powering an embedded device through a solar energy harvester has been demonstrated to be feasible and highly efficient [15]. The main challenge in such systems is the trade-off between energy management and computation, because of the non deterministic availability of solar energy. A way to address this problem is to use a solar energy predictor. Typical solar energy prediction implementations are based on *Exponential Weighted Moving Average* (EWMA) or modified EWMA [11]. There are two issues with this statistic: the predictions are done for a short time interval (e.g. 30min vs 24hrs) and the weighting factors cannot be derived with a close form solution. For these reasons we designed an algorithm capable of longer term prediction.

In our solution, the solar energy predictor estimates the total energy budget that the EH provides during the current day as a function of both the energy gathered during the previous day, which considers seasonal changes, and the current energy harvesting rate, which is affected by weather conditions. Energy income of previous day corresponds to the initial prediction of the daily energy budget, $E_P^D(0)$ (1).

$$E_P^D(0) = E_{EH}^{D-1} = \sum_{i \in D-1} e_{EH}^{D-1}(i) \cdot T_i, \quad \forall T_i \in T^{D-1} \tag{1}$$

where E_{EH}^{D-1} is the energy collected by the energy harvester during previous day, $e_{EH}^{D-1}(i)$ is the energy harvesting rate measured during the previous day at all i time slots, T_i are the time slots that divide the whole time of the day, T^{D-1}. After generating the initial estimate of the day, the algorithm adjusts the solar energy prediction, $E_P^D(t)$, by observing the behavior of the solar EH. The energy predictor compares the amount of energy collected during current day, E_{EH}^D, with the energy collected by the EH at the same time of the previous day, E_{EH}^{D-1}. These two statistics generate a prediction error correction that adjusts the daily energy budget prediction, as shown in (2).

$$\begin{aligned} E_P^D(t) &= E_P^D(0) \cdot \left[1 + \frac{E_{EH}^D(t) - E_{EH}^{D-1}(t)}{E_{EH}^{D-1}(t)} \right] \\ E_{EH}^D(t) &= \sum_{i=1}^{t} e_{EH}^D(i) \cdot T_i, \quad E_{EH}^{D-1}(t) = \sum_{i=1}^{t} e_{EH}^{D-1}(i) \cdot T_i \end{aligned} \tag{2}$$

Our algorithm considers only one day of history to build the harvesting statistics to keep the algorithm complexity low. The main objective of the energy predictor is to provide a simple and efficient estimate of daily energy availability to enable the scheduler to optimize system activity distribution over time. As the predictor updates the daily energy income, E_{EH}^D, the scheduler judges if the updated prediction is reliable or not. This prediction uses a modified version of EWMA, where two variables for automatic prediction error correction have been introduced (ε_1 and ε_2). The predictor engine (3) estimates the solar energy harvesting rate, $e_P^D(t+1)$, using the current and previous energy harvesting rates, $e_{EH}^D(t)$ and $e_{EH}^D(t-1)$ respectively, and a weighting factor, β. The weighting factor β is updated periodically according to recent irradiance statistics to follow the seasonal variance in solar irradiance. The two variables for prediction error correction, ε_1 and ε_2, are calculated as shown in (3).

$$\begin{aligned} e_P^D(t+1) &= \beta \cdot e_{EH}^D(t) \cdot (1+\varepsilon_1) + (1-\beta) \cdot e_{EH}^D(t-1) \cdot (1+\varepsilon_2) \\ \varepsilon_1 &= \frac{e_{EH}^D(t) - e_P^D(t)}{e_{EH}^D(t)}, \varepsilon_2 = \frac{e_{EH}^D(t-1) - e_P^D(t-1)}{e_{EH}^D(t-1)} \end{aligned} \tag{3}$$

System activity distribution is accomplished by equally spreading the predicted energy, E_{EH}^D, over the sequence of time intervals. This corresponds to the energy allocation per time slot available for task execution, called E_{SLOT}.

The proposed prediction engine has been implemented and verified using both real and generated solar energy profiles. Real energy profiles have been measured during 10 days of operation in the San Diego, CA [8]. The generated energy profiles belong to a finite set (sunny, cloudy, rainy day profile and either morning or afternoon cloudy and sunny for the rest of the day) and are obtained by reducing the measured solar energy irradiance by a randomly distributed noise following the patterns shown in Table 2. Each day is divided into 96 time slots of 15 minutes, which corresponds to the period between two SHM analysis in the proposed case study.

TABLE 2 WEATHER PATTERNS FOR THE VIRTUAL ENERGY PROFILES

Profile type	Generation pattern
Sunny	90-100% of max irradiance
Cloudy	70-90% of max irradiance
Rainy	50-75% of max irradiance
Cloudy morning	70-90% (Morn.) and 90-100% (Aft.) of max irradiance
Cloudy afternoon	90-100% (Morn.) and 70-90% (Aft.) of max irradiance

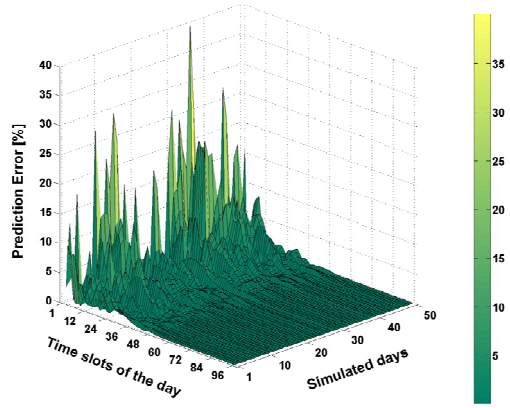

Figure 1 Prediction error evolution throughout the days

We generated a test sequence of 50 days with variable weather conditions to test the predictor accuracy. Figure 1 shows a 3D plot of the prediction error versus time slots observed in the sequence of 50 days. Prediction error drops below 10% within a 12 time slots, regardless of the different weather profiles generated.

III. PREDICTOR DRIVEN TASK SCHEDULER

Our proposed task scheduler optimizes the use of the energy allocated by the energy predictor to the current time slot. The goal is to provide a consistent level of utility of results (e.g. accuracy) over time. In structural health monitoring applications this would mean that we want to have a consistent number of measurement-analysis cycles per each time slot.

We assume that the system performs two classes of measurements: one executed periodically and a set executed on-demand with a higher priority to reflect the need for immediate results such as during catastrophic events. On-demand measurements are executed at the beginning of each time slot, and the results are sent immediately. Periodic measurements are used to build a daily statistic, which is transmitted at the end of the day. Both tasks can be executed using different DVFS modes of the CPU, leading to different power and performance profiles. In particular, we address a set of systems able to operate in two modalities: a low power, low frequency mode, and a high power, high frequency mode.

Each node works independently from the other nodes of the network. Given the reduced amount of data that needs to be sent to the backend as most analysis is performed on the sensor, and the fact that this data is sent rarely, the overhead of communication is very low to negligible. Furthermore, the nodes use low-power low-cost radios such as Zigbee. These radios can leverage ultra low power states [17] to achieve average power consumption of few mW, which can be considered negligible when compared to high performance embedded systems [8][9][16] whose power consumption is in the order of several hundred of mW. In the rest of the section we focus only on the energy consumption associated with local task execution.

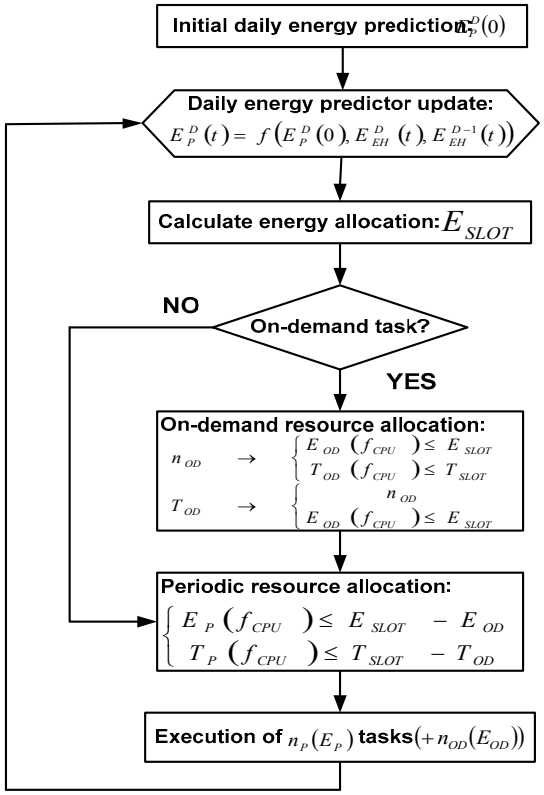

Figure 2 Predictor driven task scheduling algorithm.

Based on these assumptions, the scheduler models each task with specific energy consumption and execution time depending on DVFS mode of the CPU. Thus, the execution of multiple instances of a task consumes a predicable amount of resources. To model the resource utilization of the system, the scheduler needs to consider consumption of both periodic and on-demand tasks. Furthermore, since energy and time available are limited quantities, we assume that the system goes in a low power mode when resources for computation expire (4).

$$\begin{cases} E(t) = E_P(f_{CPU}, n_P) + E_{OD}(f_{CPU}, n_{OD}) + E_{LP} \le E_{SLOT} \\ T(t) = T_P(f_{CPU}, n_P) + T_{OD}(f_{CPU}, n_{OD}) + T_{LP} \le T_{SLOT} \end{cases} \quad (4)$$

n_P and n_{OD} are the number of repetitions of the periodic and on-demand task, respectively, while E_{LP} and T_{LP} are the energy

and time spent in *low power* mode during current time interval. The two limits, E_{SLOT} and T_{SLOT}, correspond to the limit on energy and time for task allocation. E_{SLOT} depends both on the current available energy and the predicted energy income estimated by the solar energy predictor, as discussed in the previous section. T_{SLOT} is the duration of a time interval used by the task scheduler, and depends on the monitored phenomenon.

Figure 2 shows a diagram of the proposed predictor-driven task scheduler algorithm. Once updated the energy prediction, at the beginning of each time slot, the task scheduler decides how many tasks to execute given the constraint on available energy, E_{SLOT}, and the available time, T_{SLOT}. Formally, the optimization seeks the task set S_i that, given the constraints on energy and time, maximizes an application dependent utility function, $\theta(S_i)$, which in general is a non-linear function. In this work we assume that $\theta(S_i)$ is monotonically increasing, i.e. assuming a task set $S_A \subseteq S_B$, $\theta(S_B) \geq \theta(S_A)$. In other words, as a new task is executing, the contribution to the utility of the ones already executed do not decrease.

Resource are allocated first for high priority on-demand tasks than for periodic tasks. On-demand task requests include how the system needs to perform the measurement, i.e. the needed repetitions (n_{OD}) or how long it should take (T_{OD}). Given these parameters and the amount of energy allocated for a current slot, the scheduler tries to either minimize the energy necessary to execute the required amount of repetitions (when given n_{OD}) or to maximize the number of tasks executed within the required amount of time (when given T_{OD}) (5).

$$\min\left(E_{OD} = e_{OD,f_H} \cdot n_{OD,f_H} + e_{OD,f_L} \cdot n_{OD,f_L}\right) \quad or$$
$$\max\left(n_{OD}\right)$$
$$s.t.\begin{cases} t_{OD,f_H} \cdot n_{OD,f_H} + t_{OD,f_L} \cdot n_{OD,f_L} \leq \min(T_{SLOT}, T_{OD}) & (a) \\ E_{OD} \leq E_{SLOT} & (b) \\ n_{OD,f_H} + n_{OD,f_L} = n_{OD} & (c) \end{cases} \tag{5}$$

Where n_{OD,f_H}, n_{OD,f_L} are the number of repetitions done at high and low frequency respectively, e_{OD,f_H}, e_{OD,f_L}, t_{OD,f_H}, t_{OD,f_L} are the energy and time to execute a single task either at high or low frequency. If (5) has no solution, the scheduler tries to allocate the maximum possible number of tasks given the constraints on time and energy (6).

$$\max\left(n_{OD} = n_{OD,f_H} + n_{OD,f_L}\right)$$
$$s.t.\begin{cases} e_{OD,f_H} \cdot n_{OD,f_H} + e_{OD,f_L} \cdot n_{OD,f_L} \leq E_{SLOT} & (a) \\ t_{OD,f_H} \cdot n_{OD,f_H} + t_{OD,f_L} \cdot n_{OD,f_L} \leq \min(T_{SLOT}, T_{OD}) & (b) \\ n_{OD,f_H} \geq 0 & (c) \\ n_{OD,f_L} \geq 0 & (d) \end{cases} \tag{6}$$

Once allocated the resources for the on-demand tasks, the time and energy to execute the periodic tasks are estimated (7).

$$T^*_{SLOT} = \begin{cases} T_{SLOT} & without\ \ OD \\ T_{SLOT} - T_{OD} & with\ \ OD \end{cases}$$
$$E^*_{SLOT} = \begin{cases} E_{SLOT} & without\ \ OD \\ E_{SLOT} - E_{OD} & with\ \ OD \end{cases} \tag{7}$$

The periodic task set is executed after the on-demand task. Furthermore, the results of the on-demand task are merged in the statistic of the current slot for the periodic task.

For the periodic task set we maximize the utility function $\theta(S_i)$ by calculating the optimal combination of high-frequency and low-frequency repetitions (8).

$$\max\left(\theta\left(n_{P,f_H}, n_{P,f_L}\right)\right)$$
$$s.t.\begin{cases} e_{P,f_H} \cdot n_{P,f_H} + e_{P,f_L} \cdot n_{P,f_L} + E_{LP} \leq E^*_{SLOT} & (a) \\ t_{P,f_H} \cdot n_{P,f_H} + t_{P,f_L} \cdot n_{P,f_L} + T_{LP} = T^*_{SLOT} & (b) \\ n_{P,f_H} \geq 0 & (c) \\ n_{P,f_L} \geq 0 & (d) \\ T_{LP} \geq 0 & (e) \end{cases} \tag{8}$$

where E_{LP} and T_{LP} are the energy and time consumed by the system in low-power mode, and n_{P,f_H}, n_{P,f_L} are the number of repetitions of the periodic task done at high and low frequency respectively. For each time slot, the available time end energy depends on whether an on-demand task has been executed or not. The energy consumed in low-power mode corresponds to the product of the power consumption times the time period spent in low power mode, $E_{LP} = P_{LP} \cdot T_{LP}$. Thus, in system of equations (8) we can get rid of the variable T_{LP} and obtain the problem in (9).

$$\max\left(\theta\left(n_{P,f_H}, n_{P,f_L}\right)\right)$$
$$s.t.\begin{cases} \bar{e}_{HF} \cdot n_{P,f_H} + \bar{e}_{LF} \cdot n_{P,f_L} \leq \bar{E}^*_{SLOT} & (a) \\ t_{P,f_H} \cdot n_{P,f_H} + t_{P,f_L} \cdot n_{P,f_L} \leq T^*_{SLOT} & (b) \\ n_{P,f_H} \geq 0 & (c) \\ n_{P,f_L} \geq 0 & (d) \end{cases} \tag{9}$$

where $\bar{e}_{P,f_H} = e_{P,f_H} - P_{LP} \cdot t_{P,f_H}$, $\bar{e}_{P,f_L} = e_{P,f_L} - P_{LP} \cdot t_{P,f_L}$ and $\bar{E}^*_{SLOT} = E^*_{SLOT} - P_{LP} \cdot T^*_{SLOT}$. Since $\theta\left(n_{P,f_H}, n_{P,f_L}\right)$ is monotonically increasing with the number of executed tasks (i.e. the sum of n_{P,f_H} *and* n_{P,f_L}) in problem (9) we can change the objective function to $\max\left(n_{P,f_H} + n_{P,f_L}\right)$. Therefore, problems (5), and (9) are integer linear programming (ILP) problems. These problems have solution if the constraint set, i.e. the set of the points that satisfy the equalities and inequalities, is not empty. Thus, problem (5) has solution if and only if we have enough energy to execute the on demand task within the current time slot. Similarly, problem (9) has solution if and only if $\bar{E}^*_{SLOT}$ and T^*_{SLOT} are greater than zero. Depending on the particular target application, the problems (5), and (9) can be solved using approximations [23][24], or pre-solved by embedding the solutions in look-up tables, thus having lower computational cost when executed at run-time. Linear system solution is possible if we assume that energy and time it takes to execute each task do not change. For our periodic task in (9) we want to maximize the number of tasks executed ($n_{P,f_H} + n_{P,f_L}$). Thus there is a linear relation between the energy, time and the number of task executed. Therefore, we can find an analytical solution offline and execute the task allocation with negligible computational cost.

As shown in Figure 2, at each T_{SLOT} the scheduler updates the daily energy prediction, E^D_{EH}, and consequently adjusts the amount of energy allocated per slot, E_{SLOT}. Then, if an on-

demand request has been received, the scheduler allocates and executes the n_{OD} tasks with higher priority using result of (5), or (6) if the available energy and time are too low. After executing the on-demand tasks, the scheduler allocates the n_P periodic tasks using result of (9) and the remaining system resources, $\bar{E}^*_{SLOT}$ and T^*_{SLOT}. At the end of the day, the periodic tasks results are transmitted and the scheduler builds the new prediction for the next day to restart the scheduling algorithm.

IV. Experimental Results

To validate the presented task scheduling algorithm we simulated the activity of a high performance sensor node for structural health monitoring (SHM) [16] over a period of 10 days. Energy consumption and execution time values of all tasks are obtained from power measurements on the reference device. Typical SHM analysis run has a number of averaging, filtering, feature extraction and data correlation tasks. For a single SHM analysis exploited at *high frequency* we used an average execution time of 121s and an average energy consumption of 47J; while at *low frequency* the consumption drops to 30J and the execution time rises to 148s. We compared our results with the performance obtained using both a Greedy algorithm and the scheduling algorithm for SHM applications proposed in [8] which is referred to as energy-driven algorithm. To provide the same conditions to the three algorithms, all the simulations have been performed adopting the same input power distribution, which has been measured during 10 days of operation in different weather conditions in San Diego, CA, by the solar EH implemented on the system presented in [8]. For the energy predictor we used $\beta = 0.875$.

Figure 3 shows the distribution of the allocated E_{SLOT} for three consecutive days, i.e. day 2, 3 and 4 respectively. As shown in the graph, the energy-driven algorithm allocates a higher amount of energy for system activity during the day time, sometimes reaching the system limit of 350J per T_{SLOT}, and smaller amount during the whole night time. Similarly, the Greedy algorithm consumes all the available energy during the daytime, but is not able to execute any analysis during the night time due to the lack of an energy saving policy. The proposed algorithm maintains the allocated energy quite constant maximizing the system activity over the time slots; the oscillations in the amount of allocated energy are due to prediction mistakes that occur at the beginning of the day, when the predictor assumes that the current day energy will be the same as the previous day. The last curve shows the optimal E_{SLOT} distribution calculated knowing the exact energy income of the current day; these results validate our algorithm because the optimal solution approximates the average of the allocated energy. Table 3 shows the standard deviation of the number of tasks executed in each T_{SLOT} by the different approaches. As can be seen the prediction driven approach results in approximately 5.5x reduction in standard deviation of the greedy and energy driven approaches. In fact, our algorithm's standard deviation is close to the optimal solution one.

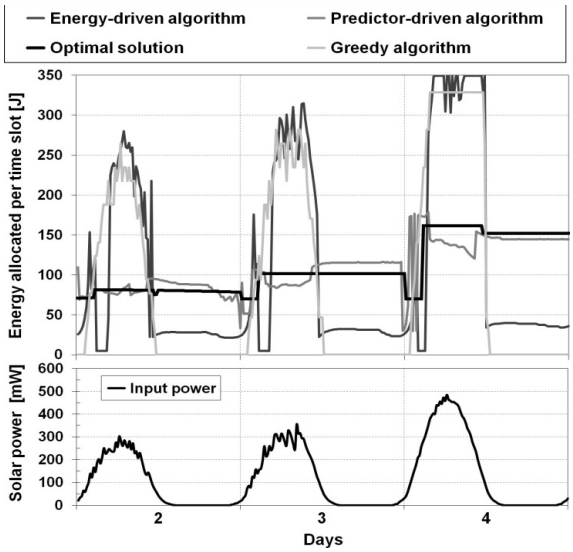

Figure 3 Example of allocated energy over 3 days

Figure 4 shows the comparison between the total numbers of monitoring tasks performed with either the proposed predictor-driven algorithm and the greedy and energy-driven algorithms for a 10 days period of continuous activity. As shown in the picture, the proposed algorithm increases the daily activity by about 50% and 30% respectively on average. Since the two algorithms are executed on the same platform and both receive the same amount of energy from the harvester, the higher number of measurements is due to better utilization of the harvested energy over time. The reason for this better energy utilization results from the even distribution of allocated energy done by the proposed energy predictor, which also improves the usage of the DVFS feature.

TABLE 3 STANDARD DEVIATION IN THE NUMBER OF MEASUREMENTS

Algorithm	Greedy	Energy driven	Predictor driven	Optimal
Standard deviation	374.5	343.1	68.8	45.0

A. Performance Evaluation Function

We defined an evaluation function that provides a metric about the effectiveness of the scheduling algorithm. The optimal solution refers to a task allocation over time that is as homogeneous as possible, hence assuring a system activity that is independent by real harvested energy distribution. Thus, the task allocation of the monitoring application can be statistically described by three indexes: 1) the average number of periodic tasks executed per time slot, avg(n_P), 2) the standard deviation of the number of periodic tasks, std(n_P), and 3) the number of on-demand tasks executed, n_{OD} (10).

$$V_{Scheduler} = k_1 \cdot \frac{avg(n_P)}{\max(n_P)} + k_2 \cdot \frac{1}{e^{std(n_P)}} + k_3 \cdot n_{OD} \qquad (10)$$

where k_1, k_2, and k_3 are weighting factors depending on the task execution priorities imposed by the monitoring application. Figure 5 shows the results obtained with the proposed function, where each curve has been normalized to the optimal solution. Simulation results show that our prediction-based task scheduling algorithm operates on average

at 80% of the ideal solution, while the energy-based algorithm reaches 55% of the optimal performance, and the Greedy approach scores only 50% due to its over aggressive approach.

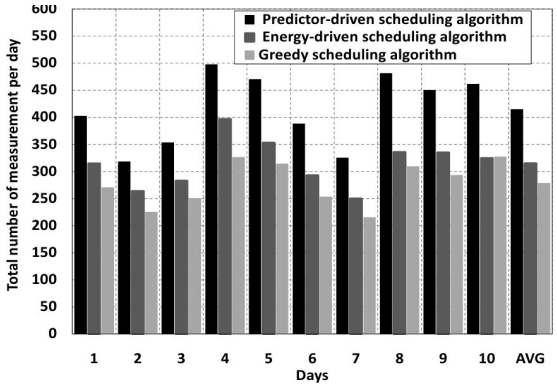

Figure 4 Total number of the data analysis performed

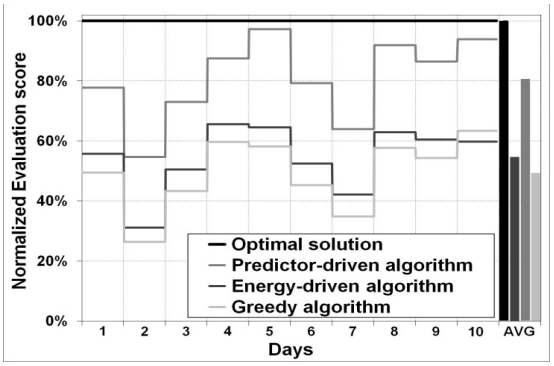

Figure 5 Value of the evaluation function

V. CONCLUSION

We presented a predictor-based task scheduling algorithm designed for applications which need to achieve a consistent level of accuracy in data monitoring and analysis independently from the solar energy availability. The proposed approach predicts the daily energy income based on location and time of the year statistic. The system allocates the predicted amount of energy for current task execution in order to provide consistent availability of resources throughout the day. Comparison of presented predictor-driven algorithm with an energy-driven and greedy scheduler showed that our algorithm is able to increase the number of daily measurements by 50% and 30% respectively while reducing the standard deviation of the number of tasks executed over time by a factor of 5.5x.

VI. ACKNOWLEDGEMENTS

This work has been founded by the NSF award number 0932403, CNS and LANL.

VII. REFERENCES

[1] Ling Q., Tian Z., Yin Y., and Li Y., "Localized structural health monitoring using energy-efficient wireless sensor networks," in IEEE sensor Journal, vol. 9 n.11, pp. 1596-1604. Nov. 2009.

[2] Wang, H., Chen, C. E., Ali, A., Asgari, S., Hudson, R. E., Yao, K., Estrin, D., "Acoustic sensor networks for woodpecker localization", in SPIE'05, vol. 5910, pp.80-91. 2005.

[3] Wnguo Y, and Tiande G., "The non-uniform porperty of energy consumption and its solution to the wireless sensor network," in ETCS'10, pp. 186-192. 2010.

[4] Biyabani A., "Infrastructure monitoring smart wireless sensor network with solar energy harvesting," in IDT'09, pp. 1-4. 2009.

[5] Cheng X., Xu J., Pei J., and Liu J.,"Hierarchical distributed data classification in wireless sensor networks,"in MASS'09, pp. 10-19, 2009.

[6] Chen B., and Tomizuka M., "Open SHM: open architecture design of Structural Health Monitoring software in Wireless Sensor Nodes," in IEEE/AME MESA'08, pp19-24. 2008

[7] Michaels J.E., Croxford A.J., and Wilcox P.D., "Imaging algorithms for location damage via in situ ultrasonica sensors," in IEEE SAS'08, pp. 63-67. 2008.

[8] Ravinagarajan A., Dondi D., and Rosing T.S., "DVFS based task scheduling in a harvesting WSN for Structural Health Monitoring", in DATE'10, pp.8-12. 2010

[9] Chen B., and Wang J., "Design of a multi-modal and high computation power wireless sesnsor node for structural health monitoring," in IEEE/ASME MESA'08, pp. 420-425. 2008.

[10] Raghunathan V., Kansal A., Hsu J., Friedman J., and Mani S., "Design consideration for solar energy harvesting wireless embedded systems," in IPSN'05, pp.457-462. 2005.

[11] Piorno J.R., Bergonzini C., Atienza D., and Rosing T.S., "Prediction and managemente in energy harvested wireless sensor nodes," in Wireless VITAE'09, pp. 6-10. 2009.

[12] Kansal A., Hsu J., Zahedi S., and Srivastava M.B., "Power management in energy harvesting sensor networks," in ACM TECS, vol.6, n. 4. Sep. 2007.

[13] Rusu C., Melhem R., and Mossè D., "Multi-version scheduling in rechargeable energy-aware real-time systems," in Journal of Embedded Computing, vol.1, n. 2, pp 271-283. Apr. 2005.

[14] Liu S., Wu Q., and Qiu Q., "An adaptive scheduling and voltage/frequency selection algorithm for real-time energy harvesting systems," in DAC'09, pp. 782-787. 2009.

[15] Dondi D., Bertacchini A., Larcher L., Pavan P., Brunelli D., and Benini L., "A solar energy harvesting circuit for low power applications," in IEEE-ICSET'08, pp. 945-949. Nov. 2008.

[16] Dondi D., Di Pompeo A., Tenti C., and Rosing T.S., "SHiMmer: a wireless harvesting embedded system for active ultrasonic structural health monitoring," in IEEE Sensors, pp. 2325-2328. Nov. 2010

[17] Benocci M., Farella E., Benini L., and Vanzago L., "Optimizing Zigbee for data streaming in body-area biofeedback applications," in 3rd IWASI'09., pp 150-155. 2009.

[18] Magno M., Brunelli D., Zappi P., Benini L., "Energy Efficient Cooperative Multimodal Ambient Monitoring," in EUROSCC'10, pp. 56-70. Nov. 2010.

[19] He Z., Wu D, "Performance analysis of wireless video sensors in video surveillance,". IEEE GLOBECOM '05, pp. 5-28. Dec. 2005.

[20] Moser C., Chen J.J, Thiele L., "Power Management in Energy Harvesting Embedded Systems with Discrete Service Levels", in ISLPED 09, pp 413-418. Aug. 2009.

[21] Moser C., Brunelli D., Thiele L., Benini L., "Real-Time Scheduling for Energy Harvesting Sensor Nodes", Real-Time Systems Journal. 2007.

[22] BTNodes. http://www.btnode.ethz.ch/

[23] Heng-Ruey Hsu, Jian-Jia Chen, and Tei-Wei Kuo, "Multiprocessor Synthesis for Periodic Hard Real-Time Tasks under a Given Energy Constraint". DATE'06, pp. 1061-1066. 2006.

[24] Yu Y, Prasanna V.K., "Power-aware resource allocation for independent tasks in heterogeneous real-time systems," ICPADS'02, pp. 341- 348. Dec. 2002.