Title: Adapting Performance in Energy Harvesting Wireless Sensor Networks for Structural Health Monitoring Applications

Authors: Jamie B. Steck
Tajana S. Rosing

ABSTRACT

SHiMmer, an active-sensing platform, combines wireless communication with solar energy harvesting to provide long-lasting structural health monitoring (SHM). While reducing the need for post-deployment physical human interaction, the platform must adapt performance to accommodate the energy availability. For example, on sunny days, SHiMmer can perform highly accurate tasks requiring more extensive computation and communication, but on cloudy days, it must reduce performance due to a decrease in harvested energy. This paper presents a system controller for SHiMmer that adapts performance based on energy availability for steady and external trigger state conditions. For steady state operation, the controller adapts the execution rate to achieve high performance while maintaining sufficient energy. For external trigger state operation, the controller determines the execution time, energy consumption and performance of a request from an external device. Using the local damage identification paradigm, the methods are tested using data from SHiMmer, showing the controller's ability to adapt at runtime and maintain sufficient energy. Steady state results show how the execution rate changes with weather conditions, while external trigger state results highlight how processing significantly affects SHiMmer's efficiency.

INTRODUCTION

SHiMmer [2], a joint development between Los Alamos National Laboratory and the University of California, San Diego, performs local structural health monitoring (SHM) through active sensing, local damage detection algorithms, and wireless communication. Active sensing occurs when actuators impart energy into the structure from which sensors measure the mechanical response [1]. SHiMmer uses piezoelectric transducers that can act as both actuators and sensors.

To reduce the need for physical human interaction, SHiMmer harvests energy

University of California, San Diego, 9500 Gilman Drive, La Jolla, CA, 92093

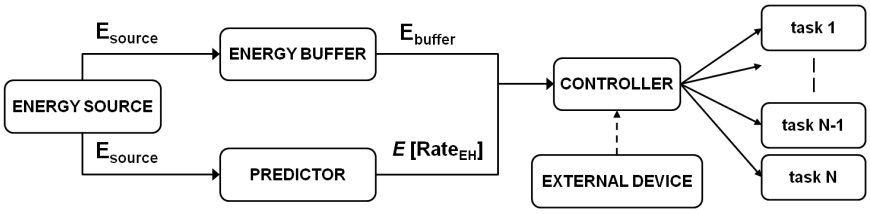

Figure 1. System Model

from the environment using a solar cell. Due to the variability of sunlight, SHiMmer must maintain energy neutrality, meaning that it can only consume as much energy as it can harvest. Typically, the goal of a sensing system is to complete a sequence of tasks within a designated period of time using a minimum amount of energy; however, to maintain energy neutrality, SHiMmer must adapt performance at runtime to exploit the characteristics of the energy source.

SHiMmer can operate in either steady state or external trigger state operation. When in steady state operation, SHiMmer periodically executes a set of tasks while maintaining energy neutrality and either sends this data to a base station or stores the results for a later time. In contrast, external trigger state operation occurs when an external device issues a request to the system that imposes a constraint. The goal of the system in this state is to meet the constraint and inform the external device of the expected performance of the request. This paper addresses the design of SHiMmer's system controller that adapts performance based on energy availability. For steady state operation, the controller adapts the execution rate to achieve high performance while maintaining sufficient energy. For external trigger state operation, the controller determines the execution time, energy consumption and performance of a request from an external device.

SYSTEM DESCRIPTION

The system model, shown in Figure 1, illustrates the components that comprise the system. The two primary software components are the predictor and the controller. The predictor uses past energy harvesting information to estimate the future rate of energy harvesting. The controller then uses this predicted information, along with the current amount of energy in the buffer, to execute a set of tasks. The following subsections describe SHiMmer's task, energy, and prediction model.

Task Model

The relationship among the tasks in a system is modeled as a Directed Acyclic Graph (DAG). In the DAG G = (**T**,**E**), each vertex represents a task $\tau \in \mathbf{T}$, which is an action or combination of actions that the system performs. Each edge $e_{ij} \in \mathbf{E}$ represents a dependency between tasks such that τ_j cannot begin execution until τ_i is complete. Each task τ is defined by its relationship with the other tasks in the system, its priority, and its application specific execution characteristics. A task τ_i can be

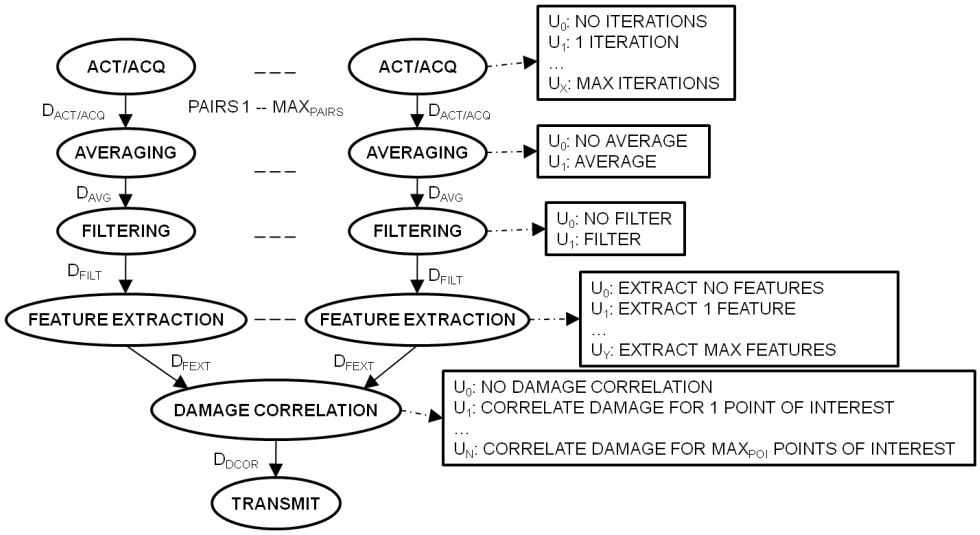

Figure 2. SHiMmer Task Graph

represented as a seven-tuple $\{p_{\tau_i}, U_{\tau_i}, t_{\tau_i}, P_{\tau_i}, s_{\tau_i}, f_{\tau_i}, d_{\tau_i}\}$ where p_{τ_i} is the priority, U_{τ_i} is the utility, t_{τ_i} is the WCET, P_{τ_i} is the power, s_{τ_i} is the start time, f_{τ_i} is the finishing time, and d_{τ_i} is the amount of data produced by the task.

During the execution of the task graph, each task τ is executed with a level of utility, U_τ, such that $0 \leq U_\tau \leq 1$. A utility of 1 represents the highest level of accuracy a task can be executed at, while a utility of 0.1 indicates little to no accuracy, and a utility of 0 indicates that the task should not be executed at all. The relative importance or priority of each task τ in the system is denoted by the task's priority, p_τ. The priority is used to determine the utility of each instance of a task in relation to other tasks in the system such that the utilities are proportional to the priorities. The expected execution time of a task is dependent on its utility. For each level of utility, the execution time of the task is referred to as $f(U_\tau)$. The energy consumption e_τ of a task τ can then be determined using the execution time t_τ and the task power P_τ.

This task model is applied to SHiMmer by creating a task graph based on local damage identification, shown in Figure 2. For each task, $f(U_\tau)$ is defined in terms of the data flow, the worst case execution time (WCET), and the power consumption. Table I shows the execution time and data formulas defined for each task, and table II shows the data used.

The first set of tasks in the graph represent actuation and acquisition. With 16 transducers, there are 120 possible one-directional pairs, where a pair is the pairing of one transducer (actuator) with another transducer (sensor). Each of the tasks $act/acq_1 - act/acq_{120}$ represents one of the 120 possible pairs that can be evaluated, the utility of which can be increased by evaluating each pair multiple times, up to $MAX_{iterations}$. Averaging, represented by tasks $avg_1 - avg_{120}$, averages multiple iterations of each pair into a single data set. If data is averaged, then the utility is 0.25 ($U = 0.25$); if data is not averaged, then the utility is 0.0 ($U = 0.0$). Filtering ($filt_1 - filt_{120}$) removes noise from the signal using a bandpass or matching filter [4][1]. If data is filtered, then the utility is 0.75 ($U = 0.75$); if data is not filtered, then the

[1]Filtering can also be done in the time domain using a moving window averaging procedure.

TABLE I. SHM EXECUTION DEFINITIONS

Task	**Execution Time: f** $(\mathbf{U}_\tau)$	**Data Produced:** d_τ	**Power:** P_τ
$act/acq_1 - act/acq_{120}$ (Actuate & Acquire)	$\mathrm{t} = (\mathrm{t_{act}} + \mathrm{t_{sense}}) * \mathrm{N_{iterations}}$ where $\mathrm{N_{iterations}} = \left\lceil \frac{e^{4.6*\mathrm{U}}-1}{\mathrm{MAX_{iterations}}} \right\rceil$	$\mathrm{d} = \mathrm{N_{iterations}} * \mathrm{d_{pair}}$	3.5 W 780 mW
$avg_1 - avg_{120}$ (Average)	$\mathrm{t} = 0 \| \mathrm{t} = \mathrm{t_{avg}} * (\mathrm{N_{iterations}} - 1)$	$\mathrm{d} = \mathrm{d}_{act/acq} \| \mathrm{d} = \mathrm{d_{pair}}$	680 mW
$filt_1 - filt_{120}$ (Filter)	$\mathrm{t} = 0 \| \mathrm{t} = \mathrm{t_{filt}} * \frac{\mathrm{d}_{avg}}{\mathrm{d_{pair}}}$	$\mathrm{d} = \mathrm{d}_{avg}$	680 mW
$fExt_1 - fExt_{120}$ (Extract Feature)	$\mathrm{t} = \lceil \mathrm{MAX_{blocks}} * \mathrm{U} \rceil$	$\mathrm{d} = \lceil \mathrm{MAX_{blocks}} * \mathrm{U} \rceil * \frac{\mathrm{d}_{avg}}{\mathrm{d_{pair}}} * \mathrm{d_{block}}$	680 mW
dCor (Correlate Damage)	$\mathrm{t} = \lceil \mathrm{MAX_{poi}} * \mathrm{U} \rceil * (\mathrm{t_{poi}} * \mathrm{N_{pairs}} + \mathrm{t_{add}} * (\mathrm{N_{pairs}} - 1))$	$\mathrm{d} = \lceil \mathrm{MAX_{poi}} * \mathrm{U} \rceil * \mathrm{d_{poi}}$	680 mW
trans (Transmit)	$\mathrm{t} = \mathrm{d}_{dCor} * \mathrm{t_{transmit/byte}} + \mathrm{t_{wakeup}}$	d = 0	168 mW

TABLE II. SHM EXECUTION DATA

$\mathrm{MAX_{iterations}}$:	10	$\mathrm{t_{act}}$:	0.1 ms	$\mathrm{t_{block}}$:	0.5596 ms	$\mathrm{d_{pair}}$:	20 KB
$\mathrm{MAX_{pairs}}$:	120	$\mathrm{t_{acq}}$:	1 ms	$\mathrm{t_{poi}}$:	0.0107 ms	$\mathrm{d_{block}}$:	2 B
$\mathrm{MAX_{poi}}$:	90	$\mathrm{t_{avg}}$:	4.004 ms	$\mathrm{t_{transmit/byte}}$:	0.071 ms	$\mathrm{d_{poi}}$:	2 B
$\mathrm{MAX_{blocks}}$:	50	$\mathrm{t_{filt}}$:	1294.6 ms	$\mathrm{t_{wakeup}}$:	13.2 ms	$\mathrm{t_{add}}$:	0.0004 ms

utility is 0.0 ($\mathrm{U} = 0.0$). Feature extraction, $fExt_1 - fExt_{120}$, divides the signal into blocks and compares up to $\mathrm{MAX_{blocks}}$ to a baseline signal. The utility for feature extraction increases as the number of features evaluated increases. Using the features from all pairs, task *dCor* performs damage correlation at a specific points of interest on the structure, up to $\mathrm{MAX_{poi}}$. Finally, task *trans* represents the transmitting of data.

Energy Model

The energy model is composed of the energy harvesting source and the energy storage buffer. The energy harvesting circuit used for SHiMmer enables the node to harvest energy from a 100 cm^2 solar cell and store up to 780 J in a 250 F, 2.5 V supercapacitor. The charging rate of the supercapacitor depends on the voltage of the solar panel, obtained from measurements further described by Recas *et al.* [3].

The energy harvested from the energy source at time t, $\mathrm{E_{harvest_t}}$, equals the amount of energy harvested between time t-1 and time t, subject to the efficiency of conversion and the energy lost due to leakage. The instantaneous rate of energy harvesting, $\mathrm{Rate_{EH}} = \frac{\mathrm{dE}}{\mathrm{dT}}$, equals the energy harvested at time t divided by the time period. Energy thresholds are set to ensure that energy neutrality is maintained. During steady

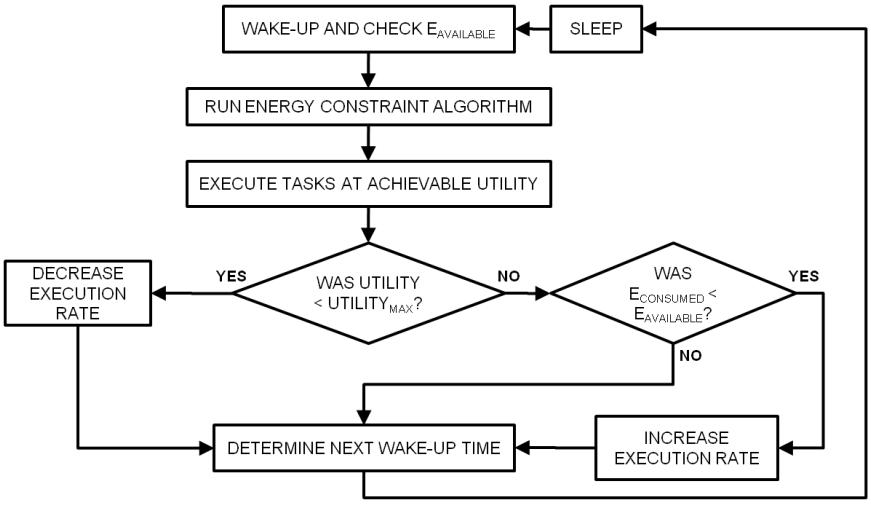

Figure 3. Steady State Control Flow

state operation, $E_{buffer} \geq E_{steady}$, and during an external request, $E_{buffer} \geq E_{min}$. The available energy at any time t is denoted as $E_{available}$.

Prediction Model

At any point in time, there may not be enough energy available to execute all the desired tasks immediately. The controller can thus estimate the additional time needed to harvest energy, t_{wait}, shown in equation 1, using the expected rate of energy harvesting, as provided by the predictor. $E_{required}$ is the sum of the energy required for each task in the system at the desired utility level.

$$t_{wait} = \frac{E_{required_t} - E_{available_t}}{E[Rate_{EH_{t+1}}]} \tag{1}$$

To estimate the amount of time needed to harvest the additional energy, the system must predict how much energy can be harvested in the future. For this, a method based on work by Recas *et al.* is used [3]. The predictor uses past energy harvesting information from previous days to estimate the rate of energy harvesting in the next period. The predictor provides a predicted rate of energy harvesting that indicates the number of expected joules per second that will be harvested in the next time period. The results from [3] show that this predictor can predict future energy within a 30 minute time frame with 10% accuracy.

STEADY STATE OPERATION

Steady state operation describes the periodic execution of tasks on a system. In some systems, steady state operation is defined by the method of duty cycling. Duty cycling, however, does not provide detailed information on the specifics of a set of tasks and assumes constant power for all tasks. Thus, for SHiMmer, steady state operation defines the periodic execution of a task graph.

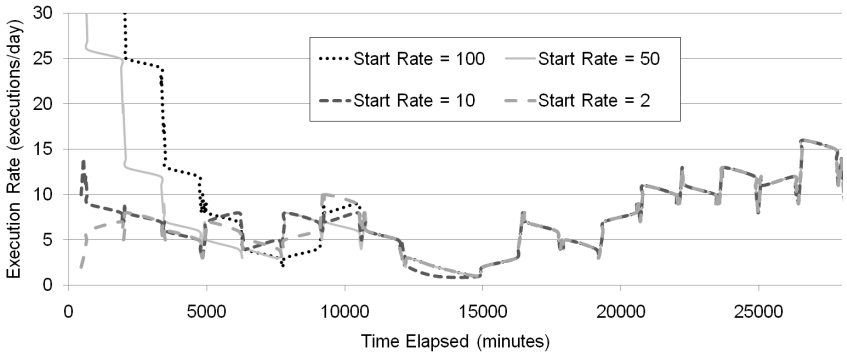

Figure 4. Convergence and Adaptation of Execution Rate

As the energy harvesting rate is variable, the rate of energy harvesting must determine how often and at what utility tasks are executed. System time is divided into $T_{positive}$, the time when energy is being harvested i.e. during the day, and $T_{negative}$, the time when energy harvesting is negligible i.e. during the night. Because $T_{positive}$ will change over time, it is updated based on previous days' positive time periods. The execution rate, r, indicates the number of times per period $T_{positive}$ that a set of tasks is executed. Thus, the task graph will be executed every t_{frame} time units, where $t_{frame} = T_{positive}/r$,. Because time period lengths and energy harvesting rates change over time, the execution rate is altered to maximize a desired parameter. For SHiMmer, the execution rate is adapted to maximize the number of maximum utility executions per period.

Figure 3 shows the flow chart for steady state operation. Every t_{frame} time units, an energy constraint algorithm[2] is executed to determine the utility of tasks given the available energy ($E_{available} = E_{buffer} - E_{steady}$). If the achievable utility is less than the maximum utility, r is decreased. If the achievable utility equals the maximum possible utility and leftover energy exists, then r is increased. The ideal situation occurs when the achievable utility equals the maximum possible utility, and the energy in the buffer equals the steady state energy threshold E_{steady}; in this case, the execution rate is not changed.

The number of maximum utility executions per day peaks at 4.55 times per day when the rate $r = 9$. When the weather patterns change or if a different set of tasks is used, however, the optimal rate will also change. The results from figure 4 show that, regardless of the initial execution rate, the rate converges and adapts to the energy profile. Specifically, days 4-9 of the evolution have a lower harvesting rate than days 10-19 (cloudy vs. sunny days) which is reflected in the increase of the execution rate. On average, the execution rate on a sunny day increases by 62% as compared to cloudy days, while the average stored energy only increases by 2%. The average stored energy remains approximately constant despite the altering weather conditions, confirming that as solar conditions change, the execution rate is adapted to maintain energy neutrality.

[2]The energy constraint algorithm determines the achievable utility given the energy available [5].

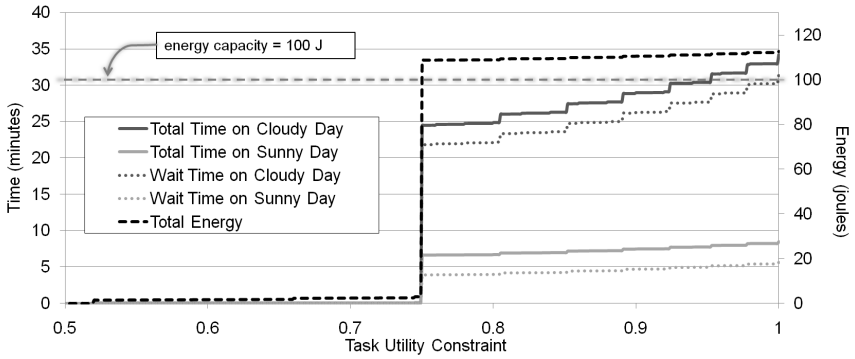

Figure 5. Comparison of Utility Requests on Sunny and Cloudy Days

EXTERNAL TRIGGER STATE OPERATION

In contrast to steady state operation, external trigger state operation occurs when an external device such as another sensor node or an unmanned vehicle initiates a request to the system [6]. Steady state operation is interrupted, and the system controller responds to the request. Because the external device does not know the energy status of the system, the system must be able to adapt to the request according to its own energy availability. The external device makes a request by specifying either a desired task utility or a maximum time limit.

Following a significant event, an external device can request results from the system indicating a required level of accuracy; in this situation, the controller runs the utility constraint algorithm[3]. The trigger specifies a desired utility for specific task(s), and the controller uses the task priorities to calculate the remaining task utilities. In contrast, data mules must typically collect data from many nodes, and thus, it may be desirable to designate a time limit in which results must be reported. When the external device specifies a time limit, the controller runs the time constraint algorithm[4] to determine a set of task utilities that satisfies the request. The system can then inform the external device of the maximum achievable utility for each task that can be executed within the given time limit, which includes the time spent harvesting additional energy.

The results of an external request depend on both the energy available and the solar conditions. The utility constraint algorithm is executed while varying the desired task utility for two distinct solar conditions: the peak of a sunny day ($\text{Rate}_{\text{EH}} = 0.1022$ mV/s) and the peak of a cloudy day ($\text{Rate}_{\text{EH}} = 0.0340$ mV/s). Figure 5 shows the energy and time requirements for tasks on both sunny and cloudy days. When the energy required to execute the tasks at the desired utility exceeds the energy available (100 J), the system waits to harvest additional energy. The wait time on a cloudy day is nearly 30 minutes greater than that on a sunny day.

In contrast to typical WSNs, SHiMmer performs complex processing on the data, dramatically reducing the amount of data transmitted. External requests with a time limit are initiated for four combinations of processing that consume various amounts

[3]The utility constraint algorithm determines the execution time of tasks at a desired utility [5].

[4]The time constraint algorithm determines the achievable utility within a given time limit [5].

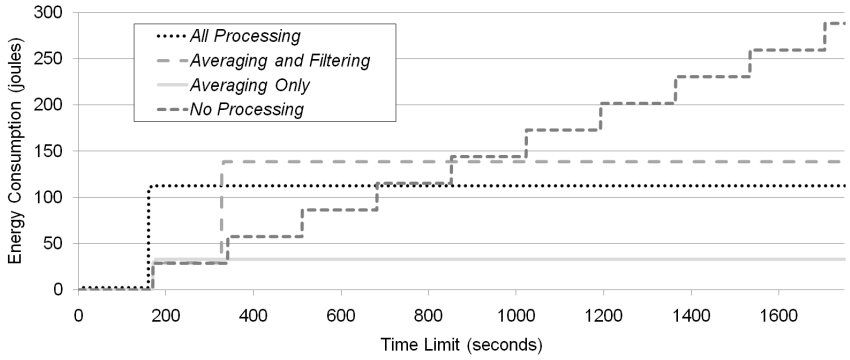

Figure 6. Energy Consumption for Various Levels of Processing

of energy and time: *all processing*, *averaging and filtering*, *averaging only*, and *no processing*. Figure 6 shows that performing *all processing* consumes less than half of the energy that *no processing* consumes, emphasizing the benefit of performing on-node data processing.

CONCLUSION

SHiMmer combines wireless communication with solar energy harvesting to provide long-lasting SHM; however, the platform must adapt performance to maintain energy neutrality. This paper described a system controller that adapts performance based on energy availability for steady and external trigger state conditions. For steady state operation, the controller adapts the execution rate to the energy profile, and for external trigger state operation, it responds to external requests according to a time or utility constraint. Using local damage identification methods and data from SHiMmer, testing of the system controller shows how energy neutrality is maintained by adapting to the energy profile.

REFERENCES

1. Drinkwater, B. W., and Wilcox, P. D., 2006: Ultrasonic arrays for non-destructive evaluation: A review. *NDT E International*, **39**(7), 525 – 541.
2. Musiani, D., Lin, K., and Rosing, T. S., 2007: Active sensing platform for wireless structural health monitoring. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, 390–399. ACM, New York, NY, USA.
3. Piorno, J. R., Bergonzini, C., Lee, B., and Rosing, T. S., 2009: Management of solar harvested energy in actuation-based and event-triggered systems. *4th Annual Energy Harvesting Workshop*.
4. Staszewski, W. J., Lee, B. C., Mallet, L., and Scarpa, F., 2004: Structural health monitoring using scanning laser vibrometry: I. lamb wave sensing. *Smart Material Structures*, **13**, 251–260.
5. Steck, J., 2009: *Energy and Task Management in Energy Harvesting Wireless Sensor Networks for Structural Health Monitoring*. Master's thesis, UCSD. Advisor - Tajana Rosing.
6. Taylor, S. G., Farinholt, K. M., Flynn, E. B., Figueiredo, E., Mascarenas, D. L., Moro, E. A., Park, G., Todd, M. D., and Farrar, C. R., 2009: A mobile-agent based wireless sensing network for structural monitoring applications. *Measurement Science and Technology*, **20**(4), 045201 (14pp).