Position Paper: Voltage Change is not Energy Consumption

Pouya Mahdi Gholami*
pouya@uchicago.edu
University of Chicago
Chicago, IL, USA

Mingyuan Xiang*
myxiang@uchicago.edu
University of Chicago
Chicago, IL, USA

Henry Hoffmann
hankhoffmann@uchicago.edu
University of Chicago
Chicago, IL, USA

Abstract

Energy-harvesting sensors utilize local, ambient energy resources to operate and thus eliminate the need for batteries. A key challenge for such systems is avoiding power failures during application execution. Energy-aware runtimes avoid such failures by reasoning about the task's energy consumption and the current energy available to the system. However, the energy consumption estimates profiled by prior work fail to account for incoming energy, producing incorrect energy consumption estimates which could lead to power failures and missed deadlines. This work analyzes the impact of incoming energy on the profiled energy consumption and argues that future energy-aware runtimes must be mindful of harvested energy when profiling a task's energy consumption.

CCS Concepts

• **Computer systems organization** → **Embedded software**.

Keywords

Intermittent Computing, Energy-Harvesting Power System, Energy Estimation

ACM Reference Format:
Pouya Mahdi Gholami, Mingyuan Xiang, and Henry Hoffmann. 2025. Position Paper: Voltage Change is not Energy Consumption. In *The 13th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems (ENSsys '25), May 6–9, 2025, Irvine, CA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3722572.3727932

1 Introductions

Progress in energy-harvesting, battery-free technology has given rise to sensing and processing applications on the edge [1–7, 9, 10, 12–14, 16, 16, 18–20]. These systems collect ambient energy from environmental sources such as solar, radio frequency, thermal or vibrations and store it in capacitors. When the stored energy is sufficient, intermittent sensors can execute sensing, processing, or communication tasks. When energy is insufficient, the system turns off to recharge until sufficient energy becomes available. As a consequence, typical intermittent execution is frequently hindered by *power failures*, which occur when the capacitor's voltage falls below the harvesting hardware's minimum threshold (V_{off}). To support the intermittent execution and safely tolerate power failures, prior work has explored manual or automatic system state checkpointing [2, 10, 16, 19], reformatting applications into atomic, task-based execution models [5, 12, 20], or a combination of the two approaches [11, 14, 18].

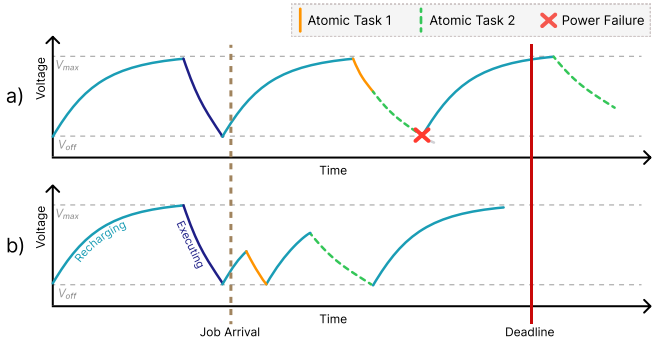

Figure 1: Energy-aware solutions avoid energy wastage and potential deadline misses. This figure compares the execution of a job consisting of two atomic tasks between the job's arrival and its deadline under an energy-aware and a traditional runtime. a A traditional runtime charges the system until the capacitor voltages reaches V_{max} and executes tasks until the voltage reaches V_{off}. This mode of operation causes power failures, wasting valuable energy, and can miss job deadlines. b) Energy-aware runtimes monitor the system's energy and execute atomic tasks when sufficient energy is present in the system, avoiding power failure and meeting the task's deadline.

As explored by prior work, timely scheduling of atomic tasks are a fundamental challenge for intermittent runtime systems [7–9, 15, 18, 20]. Atomic tasks—such as sampling via peripherals or radio communication—must be executed without interruptions *e.g.* power failures. If power failures occur during an atomic task, the intermittent runtime must atomically re-execute the entirety of the task again. Many prior systems [11, 12, 14, 20] support such atomic tasks by allowing for task re-execution while preventing potential memory corruption and non-termination. While atomic task re-execution is a functionally correct method in the face of frequent power failures, it wastes precious energy when such tasks are inevitably interrupted by power failures.

Energy-aware solutions circumvent the wasted energy issue by avoiding power failures during task execution [4, 8, 15, 17]. First, these runtime estimate a task's energy consumption (E_{task}) and the system's available energy (E_{cap}) via $E_{\text{cap}} = \frac{1}{2}CV_{\text{cap}}$, where V_{cap} is the capacitor's voltage and C is its capacitance. Then, throughout their operation these systems execute a task only if E_{cap} is sufficient to ensure that the system capacitor's voltage remains above V_{off} throughout the task's operation, *i.e.* a task is executed only when

*Both authors contributed equally to this research.

ENSsys '25, May 6-9, 2025, Irvine, CA, USA

ACM ISBN 979-8-4007-1606-5/2025/05
https://doi.org/10.1145/3722572.3727932

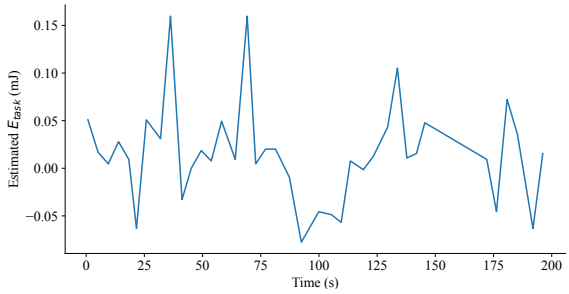

Figure 2: E_{task} of an atomic microphone sampling task on a radio frequency harvesting sensor, estimated approximately every 5 seconds across 200 seconds. Task energy was estimated using Equation 2. While the position of the sensor was not changed (i.e. its distance to the broadcaster was constant), the foot traffic in the office resulted in different amounts of energy harvested by the node during task profiling, resulting in inconsistent E_{task} estimates.

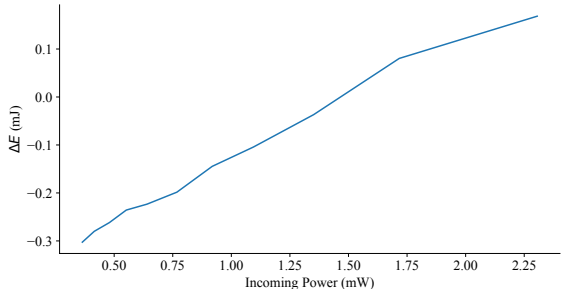

Figure 3: ΔE of an atomic microphone sampling task as the incoming power to the harvester changes. When incoming power is high, more energy is harvested than spent during the task's execution, so $V_{\text{post}} > V_{\text{pre}}$ and $\Delta E > 0$. Inversely, when incoming power is low, less energy is harvested than spent during the task's execution and $\Delta E < 0$.

$E_{\text{cap}} - E_{\text{task}} > E_{\text{off}}$, where E_{off} represents the energy present in the capacitor at V_{off}. Figure 1 depicts how an energy-aware runtime (b) conserves energy and avoids missed deadlines when compared to traditional runtimes (a).

Estimated E_{task}s play a central in the operation of these runtimes: E_{task} is used to ensure power failures do not occur, to measure the feasibility of the current schedule, and to appropriately degrade performance when the current schedule is not feasible, and to respond to reactive events in a timely manner [15]. To attain the benefits of such systems, energy-aware runtimes need accurate estimates of a task's energy consumption (E_{task}) under the energy harvesting architecture [15, 17]. To obtain such estimates, these efforts need to profile a task's energy consumption and **assume that future execution of a task will behave similarly to the profiled iteration of the task**, *i.e.* a future version of the task will consume the same amount of E_{task}. However, current profiling approaches [15, 17] lead to inconsistent E_{task} approximations.

During profiling, energy-aware runtimes estimate the energy change during a task as ΔE and measure it via

$$\Delta E = \frac{1}{2}C(V_{\text{post}}^2 - V_{\text{pre}}^2) \tag{1}$$

, where V_{pre} and V_{post} correspond to the capacitor's voltage before and after the task's execution [15, 17]. They then attribute most of change in energy to the task's execution and estimate E_{task} as

$$\Delta E \approx -E_{task} \tag{2}$$

However, based on our experiments, Equation 2 leads to inconsistent E_{task} estimates. Figure 2 depicts 40 E_{task} estimates of the same task across 200 seconds on a radio frequency harvesting node deployed in an office. As shown, the E_{task} estimates vary wildly, ranging between -0.05 and 0.15 mJ[1].

After careful reconsideration of the experiment, **the energy harvested during the task is determined as the source of the inconsistency**. To test this hypothesis, we measure ΔE based on 1 while changing the incoming power into the harvester. The results are shown in figure 3. As shown, ΔE is directly impacted by the incoming power of the system, implying that the ΔE depends not only on the energy consumed by the task, but also on the energy harvested during the task. Hence, we reformulate Equation 2 as follows,

$$\begin{aligned} \Delta E &= E_{\text{harv}} - E_{\text{task}} \\ E_{\text{task}} &= E_{\text{harv}} - \frac{1}{2}C(V_{\text{post}}^2 - V_{\text{pre}}^2) \end{aligned} \tag{3}$$

where E_{harv} is the energy harvested during the task's execution. To accurately profile E_{task}, runtimes need to also accurately estimate E_{harv}. However, measuring E_{harv} independently of E_{task} during a task's execution is an open problem.

One approach to handling this challenge is to assume that E_{harv} is constant as long as energy conditions remain stable and continue using Equation 2. When energy conditions do change, this approach simply re-profiles ΔE to compensate for the changed E_{harv} [17]. However, prior work has not explored how much deviation from stable conditions would require re-profiling. Further, this approach assumes that incoming power changes slowly enough such that the re-profiling is seldom needed and when needed, there is ample time and energy for re-profiling. Such an approach would fail in the case of arbitrarily or quickly changing energy conditions.

A potential solution would be to directly estimate E_{harv}. Let P_{in} denote the power incoming to the system and T_{task} the duration of a task. Then, assuming P_{in} is stable, harvested energy can be described as $E_{\text{harv}} = P_{\text{in}} \cdot T_{\text{task}}$. However, this approach would also fail in the case of arbitrarily or quickly changing energy conditions. Additionally, given the non-linear rate of charging often seen in intermittent systems, it is not clear whether P_{in} can be accurately measured even during stable energy conditions.

This work demonstrates that prior work assumes that the actual energy consumed by a task (E_{task}) can be directly approximated using the profiled change in the capacitor's energy (ΔE) the task. We argue and show that this assumption can lead to inconsistent

[1]Negative estimates occur when $V_{\text{post}} > V_{\text{pre}}$, *i.e.* when the system has gained more energy during a task's execution.

E_{task} estimates, which – by assumption – should be constant. We experimentally show that the inconsistency stems from the energy harvested during the task (E_{harv}). Finally, we argue that to maintain the benefits of energy-awareness for all incoming energy conditions, future energy-aware runtime must approximate E_{task} based on both ΔE and E_{harv}.

2 Background on Energy Aware Scheduling

Energy-aware runtimes estimate a task's E_{task} either via dynamic profiling [15, 17] or statically-guided modeling [8, 17], though dynamically profiled approaches are more accurate and can operate with any energy-harvesting hardware. Statically-guided energy cost estimations range from simple to sophisticated. Simple static estimations measure the E_{task} on wall power and assume that intermittent execution of the task will behave similarly [8]. However, these estimations do not account for the inefficiencies of energy-harvesting circuits [17]. Sophisticated solutions aim to resolve this issue by modeling the energy-harvesting hardware to provide more accurate E_{task} estimations [17]. While more accurate, the improved accuracy of these solutions is directly tied to the accuracy of the efficiency model and compounding errors in the model can lead to increasingly inaccurate estimates [17]. Further, changing any energy-harvesting component would require updating the model. In contrast, dynamically profiled methods estimate E_{task} by measuring the capacitor's voltage before (V_{pre}) and after (V_{post}) and using Equation 2. [15, 17]. This model-free dynamic approach includes the inefficiencies of the system, producing more accurate results than static approaches with limited reasoning about the energy-harvesting hardware [17].

Acknowledgments

We also thank the anonymous reviewers for their feedback. Funding for this work comes from the National Science Foundation (CCF-2119184, CNS-2313190, CCF-1822949 and CNS-1956180).

References

[1] Abu Bakar, Alexander G. Ross, Kasim Sinan Yildirim, and Josiah Hester. 2021. REHASH: A Flexible, Developer Focused, Heuristic Adaptation Platform for Intermittently Powered Computing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 3 (Sept. 2021), 87:1–87:42. doi:10.1145/3478077

[2] Domenico Balsamo, Alex S. Weddell, Geoff V. Merrett, Bashir M. Al-Hashimi, Davide Brunelli, and Luca Benini. 2015. Hibernus: Sustaining Computation During Intermittent Supply for Energy-Harvesting Systems. *IEEE Embedded Systems Letters* 7, 1 (March 2015), 15–18. doi:10.1109/LES.2014.2371494 Conference Name: IEEE Embedded Systems Letters.

[3] Fulvio Bambusi, Francesco Cerizzi, Yamin Lee, and Luca Mottola. 2022. The Case for Approximate Intermittent Computing. In *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 463–476. doi:10.1109/IPSN54338.2022.00044

[4] Naveed Anwar Bhatti and Luca Mottola. 2017. HarvOS: Efficient Code Instrumentation for Transiently-Powered Embedded Sensing. In *2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 209–220. https://ieeexplore.ieee.org/abstract/document/7944791

[5] Alexei Colin and Brandon Lucia. 2016. Chain: tasks and channels for reliable intermittent programs. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*. ACM, Amsterdam Netherlands, 514–530. doi:10.1145/2983990.2983995

[6] Jasper de Winkel, Carlo Delle Donne, Kasim Sinan Yildirim, Przemysław Pawełczak, and Josiah Hester. 2020. Reliable Timekeeping for Intermittent Computing. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 53–67. doi:10.1145/3373376.3378464

[7] Josiah Hester, Kevin Storer, and Jacob Sorber. 2017. Timely Execution on Intermittently Powered Batteryless Sensors. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (SenSys '17)*. Association for Computing Machinery, New York, NY, USA, 1–13. doi:10.1145/3131672.3131673

[8] Bashima Islam and Shahriar Nirjon. 2020. Scheduling Computational and Energy Harvesting Tasks in Deadline-Aware Intermittent Systems. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 95–109. doi:10.1109/RTAS48715.2020.00-14 ISSN: 2642-7346.

[9] Bashima Islam and Shahriar Nirjon. 2020. Zygarde: Time-Sensitive On-Device Deep Inference and Adaptation on Intermittently-Powered Systems. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 3 (Sept. 2020), 82:1–82:29. doi:10.1145/3411808

[10] Hrishikesh Jayakumar, Arnab Raha, and Vijay Raghunathan. 2014. QUICKRECALL: A Low Overhead HW/SW Approach for Enabling Computations across Power Cycles in Transiently Powered Computers. In *2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*. 330–335. doi:10.1109/VLSID.2014.63 ISSN: 2380-6923.

[11] Vito Kortbeek, Kasim Sinan Yildirim, Abu Bakar, Jacob Sorber, Josiah Hester, and Przemysław Pawełczak. 2020. Time-sensitive Intermittent Computing Meets Legacy Software. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 85–99. doi:10.1145/3373376.3378476

[12] Kiwan Maeng, Alexei Colin, and Brandon Lucia. 2017. Alpaca: intermittent execution without checkpoints. *Proceedings of the ACM on Programming Languages* 1, OOPSLA (Oct. 2017), 96:1–96:30. doi:10.1145/3133920

[13] Kiwan Maeng and Brandon Lucia. 2018. Adaptive dynamic checkpointing for safe efficient intermittent computing. In *Proceedings of the 13th USENIX conference on Operating Systems Design and Implementation (OSDI'18)*. USENIX Association, USA, 129–144.

[14] Kiwan Maeng and Brandon Lucia. 2019. Supporting peripherals in intermittent systems with just-in-time checkpoints. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2019)*. Association for Computing Machinery, New York, NY, USA, 1101–1116. doi:10.1145/3314221.3314613

[15] Kiwan Maeng and Brandon Lucia. 2020. Adaptive low-overhead scheduling for periodic and reactive intermittent execution. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2020)*. Association for Computing Machinery, New York, NY, USA, 1005–1021. doi:10.1145/3385412.3385998

[16] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2011. Mementos: system support for long-running computation on RFID-scale devices. *ACM SIGARCH Computer Architecture News* 39, 1 (March 2011), 159–170. doi:10.1145/1961295.1950386

[17] Emily Ruppel, Milijana Surbatovich, Harsh Desai, and Brandon Lucia. 2022. An Architectural Charge Management Interface for Energy-Harvesting Systems. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, Chicago, IL, USA, 318–335. doi:10.1109/MICRO56248.2022.00034

[18] Milijana Surbatovich, Limin Jia, and Brandon Lucia. 2021. Automatically enforcing fresh and consistent inputs in intermittent systems. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI 2021)*. Association for Computing Machinery, New York, NY, USA, 851–866. doi:10.1145/3453483.3454081

[19] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent computation without hardware support or programmer intervention. In *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation (OSDI'16)*. USENIX Association, USA, 17–32.

[20] Kasım Sinan Yıldırım, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemyslaw Pawelczak, and Josiah Hester. 2018. InK: Reactive Kernel for Tiny Batteryless Sensors. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (SenSys '18)*. Association for Computing Machinery, New York, NY, USA, 41–53. doi:10.1145/3274783.3274837