

Towards Efficient Offloading of Time-Critical Tasks in Autonomous Edge-to-Fog Paradigms

Rola El Osta
LENS Laboratory
Lebanese University
Saida, Lebanon
rola.elosta@ul.edu.lb

Maryline Chetto
0000-0003-1118-2279
Nantes Université, École Centrale Nantes
CNRS, LS2N, UMR 6004, F-44000 Nantes, France
maryline.chetto@univ-nantes.fr

Abstract—The uncontrollable and time-varying nature of harvested energy makes it challenging to guarantee timing requirements of the real-time software in Energy-Harvesting (EH) autonomous edge devices. In this paper, we consider that the autonomous edge device supports two kinds of hard deadline tasks: low computational periodic tasks in charge of sensing and high computational aperiodic tasks in charge of AI models. This paper provides an algorithm for online acceptance testing that decides whether to process the aperiodic task locally or offload it to the fog, based on current energy and processing conditions. This research represents a foundational step in the integration of fog resources into the IoT infrastructure, bridging the gap between local autonomy and distributed computing.

Index Terms—Autonomous edge device, real-time computing, energy harvesting, offloading, scheduling, acceptance test, Earliest Deadline First.

I. INTRODUCTION

IN THE IoT infrastructure, the edge device integrates computation and networking facilities so as to monitor and even control physical processes with feedback loops as in the so-called CPSs [1]. It carries out low complexity algorithms such as data sorting and filtering. Data are gathered from sensors and other sources within the edge device to serve as input of AI models for prediction, classification or control [2]. Implementing Artificial Intelligence (AI) models then leads to smart decision-making without human intervention. The aim of the fog computer is to permit further analysis of data issued from the edge device when it does not have enough processing power or/and energy to handle the computations itself [3].

In healthcare, AI models analyze data coming periodically from wearable devices and electronic health records in order to predict potential health issues, enabling early intervention and personalized care. Real-time patient monitoring hence provides immediate insights for timely interventions, improving patient outcomes. In industrial predictive maintenance, AI algorithms predict equipment failures before they occur, allowing for timely maintenance and reducing downtime. Clearly, the edge device supports a set of periodic tasks in charge of cyclically collecting data while aperiodic AI tasks analyze data for decision-making [4] [5].

However, an edge device may be limited by its processing power which makes it challenging to run complex AI tasks efficiently. Achieving responses within stringent timing re-

quirements is crucial in critical applications such as healthcare [6]. The edge device may also be limited in terms of energy when it harnesses ambient energy from its environment for its supply. Solar, thermal and kinetic are typical environmental energy sources. An Energy Harvesting (EH) device does not need for wired power connections and battery replacements which makes it possible perpetual operation through an adequate operational mode known as energy neutral. An EH edge device incorporates ultra low-power micro-controllers which complicate deployment of large AI models. In most sensing devices, the application software is composed of hard deadline constrained tasks which may be either periodic or aperiodic. For example, in a vital sign monitoring system, the wearable device periodically measures heart rate, blood pressure, and oxygen saturation. AI tasks analyze these readings in order to detect anomalies or predict potential health issues. If a sudden irregular heartbeat is detected, an aperiodic AI task is activated to evaluate the condition and alert medical personnel if necessary. Another example is transportation where autonomous sensors on roads collect data on traffic conditions periodically and AI tasks use this data to optimize traffic signals and reduce congestion.

A strategic approach hence has to balance variable computational demands of the tasks with uncontrollable and time-varying power availability. While the periodic sensing tasks have low complexity, AI tasks typically have high computation times required by accurate decision-making. This makes it hard to guarantee beforehand their timely execution in a resource-constrained device such as an EH sensor node. This involves integrating a real-time energy-aware scheduler combined with dynamic acceptance testing and task offloading to fog. Fog nodes offer greater computational resources than edge devices. Offloading AI tasks with intensive computation from the edge to the fog will permit to guarantee schedulability whenever possible. The task offloading decision supposes to implement the so-called acceptance test to decide whether to offload the task or to execute it locally. If offloaded, the task is transmitted to a suitable fog node which processes the task and returns the results to the edge device. In this paper, we assume the optimal energy-aware ED-H for local scheduling on the EH edge device. Two kinds of tasks are considered: low computational periodic tasks initially and

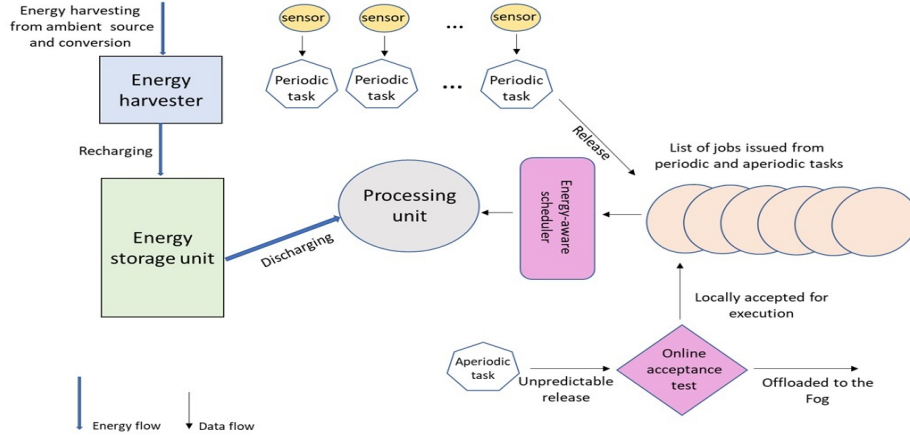


Fig. 1: Infrastructure of the sustainable edge device under study.

definitively assigned to the node (sensing, filtering) and high computational aperiodic tasks (AI models) that need to be guaranteed at runtime through acceptance testing to decide whether they are candidate for offloading. We propose an efficient online acceptance test that determines the schedulability of incoming aperiodic tasks. The test is based on a slack stealing mechanism for time and energy. It guarantees that the deadlines of all locally executed tasks may be met. To the best of our knowledge, this is the first work to deal with this issue under both time and energy harvesting settings. We do not focus on the dispatching strategy in charge of assigning the offloaded tasks (i.e. the locally rejected tasks) to appropriate fog nodes. In that objective, the dispatcher generally has to consider factors such as proximity, energy availability and current processing load.

The remainder of this paper is structured as follows: The subsequent section delineates the system model under study. Section 3 addresses the principle of online acceptance testing. Section 4 presents the simulation results. Section 5 discusses the principles underlying fault-tolerant offloading. Section 6 reviews related works in the field. Finally, Section 7 summarizes the conclusions of this study.

II. PROBLEM STATEMENT AND MODEL

The system has four parts: an energy source, a harvester, an energy storage unit and a uniprocessor computing unit. The harvester draws energy from the environmental source and transforms the energy into electrical power. Electrical power is then stored in capacitor or/and rechargeable battery from the harvester. Finally, the uniprocessor system receives energy from the storage unit with fluctuant energy availability.

A. Energy and time model

The energy source model is characterized by $Pr(t_c)$ which denotes the replenishment function that is actually fed into the energy storage. The energy storage unit is replenished even during task execution. We use $E(t)$ to denote the amount of energy stored in the energy storage unit at time t_c . A

number of periodic tasks are statically assigned to the single processing unit which operates at a fixed CPU frequency. The operating system provides preemptive priority-based dispatching using the Earliest Deadline First priority assignment rule. We consider a set of n independent and preemptable periodic tasks described as follows: $\tau = \{\tau_i | 1 \leq i \leq n\}$. A four-tuple (C_i, D_i, T_i, E_i) characterizes the task τ_i where C_i , D_i , T_i and E_i indicate the worst case execution time, the relative deadline, the period and the worst case energy consumption of τ_i , respectively. We assume that the task set exhibits no synchronization and that each invocation of a task takes its worst case execution time and its worst case energy consumption [7]. At a given time t_c , the periodic task set is characterized by all the jobs generated by τ which are not completed at time t_c and jobs which will be released after time t_c . The average ratio of time where the processor is busy for executing jobs issued from τ is no more than one. The average power it consumes is no more than the average power produced by the environmental source. We assume that the periodic tasks may be feasibly executed even in the worst case energy situation.

Aperiodic tasks are event-driven and occur irregularly at unpredictable times. One example of aperiodic action to be performed timely is event detection and reporting. The aperiodic task has to respond to a sudden environmental change such as detecting motion in a secured area and sending an immediate alert. Thus, an aperiodic task J_i has arrival time a_i , worst case execution-time c_i , worst case energy consumption e_i and absolute deadline d_i . It is assumed that the execution time, energy consumption and deadline of the arrived aperiodic tasks are known upon arrival.

B. Background on the ED-H scheduler

ED-H is not work-conservative since the processor can be put in the sleep mode deliberately even if one or more tasks are pending for execution. The scheduler ED-H uses two fundamental online data:

- the slack time of the periodic task set τ at t_c , $ST_\tau(t_c)$ gives the maximum continuous time that could be made available from t_c by tasks of τ while still guaranteeing no deadline missing for τ . $ST_\tau(t_c)$ is obtained from the Earliest Deadline as Late as possible (EDL) schedule as described in [8].
- The preemption slack energy of a periodic task set τ at t_c , $PSE_\tau(t_c)$ gives the maximum energy that could be consumed from time t_c by the highest priority task ready to execute while still guaranteeing no deadline missing for future tasks of τ .

The processor management strategy of ED-H is the following: the processor is busy if the energy storage unit is below a specified threshold or the system has no preemption slack energy. In contrast, the processor is idle if the energy storage unit is fully replenished or the system has no slack time. In any other situation, the processor may be busy or idle. We may opt for a strategy that we may call the ASAP-ALAP version of ED-H: The processor starts executing a job (end of recharging phase) only when the storage unit has fully replenished and the processor stops executing jobs (beginning of recharging phase) when the storage unit has fully depleted. Under the ASAP-ALAP version of ED-H, the electronic circuits spend as much time as possible in the low-power state before transitioning to the active operating mode. As a consequence, energy overheads as well as time overheads due to switching between different power modes are minimized.

We illustrate the ED-H scheduler on a periodic task set composed of two tasks for simplicity, as given by Table I. We assume that the energy storage capacity is equal to 20 mJ and $E_{min} = 2$. The rechargeable power is constant along time and equals 3 mW. An aperiodic task J_1 arrives at time 3 with absolute deadline at time 7.

Periodic task	C_i (s)	D_i (s)	T_i (s)	E_i (mJ)
τ_1	1	5	8	10
τ_2	1	8	8	8
Aperiodic task	C_i (s)	D_i (s)	a_i (s)	E_i (mJ)
J_1	3	7	3	12

TABLE I: Time and energy parameters of the tasks

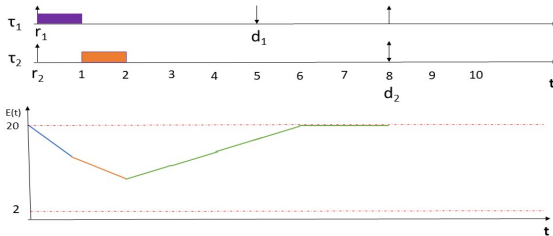


Fig. 2: Illustration of the ED-H energy-aware scheduler

The job τ_1 is released at time 0. As there is energy in the storage unit, τ_1 executes until $t=1$. When the other job τ_2 releases, there is sufficient energy to complete it by its deadline. It executes until it is suspended to make the processor idle and allow the storage unit to recharge. The intuition of ED-H is to consider future arrivals of jobs as well as their energy requirements to make dispatching decision i.e. to decide whether or not putting the processor in the sleep mode. The example in Figure 4 illustrates this fact. A good scheduler should avoid energy starvation whenever possible. ED-H examines not only the energy that is currently stored in the storage unit but also the energy that will be produced by the environmental source and consumed by tasks in future.

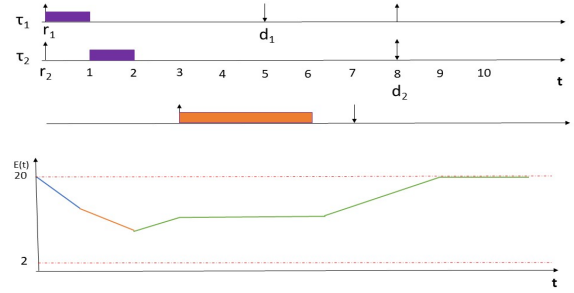


Fig. 3: Illustration of the ED-H energy-aware scheduler with no acceptance testing

Assume now that no acceptance test is available (Figure 3) That aperiodic task that occurs is automatically accepted and scheduled jointly with the periodic tasks on a best-effort principle. Obviously, some processing overload situations may occur as well as energy starvation situations. The consequences will be deadline missings for the tasks, periodic and/or aperiodic ones, in a nondeterministic manner.

III. ONLINE ACCEPTANCE TESTING

A. Issue under study

We assume that the device has to support a set of predefined periodic tasks and unpredictable occurring aperiodic tasks. All the tasks, once accepted are jointly scheduled according to ED-H, the optimal energy harvesting aware algorithm. Let $J_c(c, e, d)$ be the new occurring aperiodic task. Acceptance of J_c is tested once at the time of its arrival, say time t_c . The acceptance test gives a definitive answer regarding the question "Can J_c be accepted?". A positive answer signifies that adding J_c will not result in deadline violation for any job generated by the periodic tasks in τ and any previously accepted aperiodic task. Under EH settings, a task can miss its deadline because of time starvation or energy starvation: Time starvation is when a task reaches its deadline at time t , its execution is incomplete because the time required to process it by its deadline is not sufficient. There is available energy in the storage unit when the deadline violation occurs (i.e. $E(t) > 0$).

Energy starvation is when the task reaches its deadline at time t , its execution is not complete because the energy required to process it by its deadline is not available. The energy in the storage is exhausted when the deadline violation occurs (i.e. $E(t) = E_{min}$).

B. Principles

Our acceptance test is based on a slack stealing technique, designed to improve the responsiveness of aperiodic tasks without compromising the deadlines of periodic tasks possibly coming from energy starvation or processing overload. Periodic tasks have well known execution patterns and well known energy consumptions. Periodic tasks do not utilize 100 percent of the processing capacity and 100 percent of the available energy. This results in unused processor time and unused energy which could serve to execute the aperiodic tasks. Consequently, we extend the slack stealing approach which initially, only concerned slack time. The idea is to intelligently utilize both slack time and slack energy to accommodate the hard aperiodic tasks, ensuring that all tasks meet their deadlines.

Let J_i be a hard aperiodic task with absolute deadline d_i , execution time C_i and energy consumption C_i . Let us introduce $\delta_i^t(t_c)$ called the dynamic time laxity of J_i at time t_c , given by $\Omega_\tau(t_c, d_i) - \sum_{d_j \leq d_i} c_j$. $\Omega_\tau(t_c, d_i)$ represents the maximum processing surplus which may be made available within the scheduling interval of J_i after guaranteeing that all periodic tasks meet their deadline and assuming that the periodic tasks are scheduled according to the Earliest Deadline First rule. $\sum_{d_j \leq d_i} c_j$ gives the total execution time required by the aperiodic tasks with a priority higher than or equal to J_i at time t_c .

Identically, we introduce $\delta_i^e(t_c)$ called the dynamic energy laxity of J_i at time t_c , given by $\Pi_\tau(t_c, d_i) - \sum_{d_j \leq d_i} e_j$. $\Pi_\tau(t_c, d_i)$ is given by $E(t_c) + E_p(t_c, d_i) - g_p(t_c, d_i)$. $E(t_c)$ is the residual capacity of the energy storage unit at t_c . $E_p(t_c, d_i)$ is the total energy produced by the environmental source between time t_c and time d_i . $g_p(t_c, d_i)$ refers to the energy demand issued from the periodic tasks between t_c and d_i . Therefore, $g_p(t_c, d_i) = \sum_{j=1}^n \lceil \frac{d_i - t_c}{T_j} \rceil E_j + \sum_{k=1}^n (d_k > d_i) E_k(t)$. Here, $E_k(t)$ denotes the remaining amount of energy of the current invocation of the periodic task τ_k at time t_c .

The following necessary and sufficient condition has been drawn for the acceptance test:

Proposition 1: Aperiodic task J_c with deadline d is accepted if and only if for every aperiodic task J_i such that $d_i \geq d$,

$$\delta_i^t(t_c) \geq 0 \text{ and } \delta_i^e(t_c) \geq 0 \quad (1)$$

C. Illustrative example

In Figure 3, the periodic task set τ is scheduled according to ED-H till time $t = 3$ where $E(3) = 11$ mJ. J_1 is occurring

at time 3. The acceptance test requires to compute $\delta_1^e(3)$ and $\delta_1^t(3)$. $\delta_1^e(3) = E(3) + \int_3^7 -e_1 = 11 > 0$ and $\delta_1^t(3) = 1 > 0$. As both the time laxity and energy laxity of J_1 are positive, J_1 is accepted.

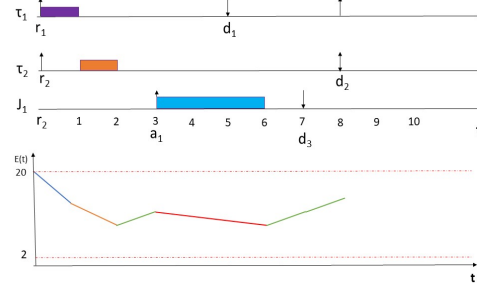


Fig. 4: Illustration of the Slack Stealing based acceptance test

IV. SIMULATION RESULTS

To evaluate the performance of the acceptance test, we developed a simulator to run a task set composed of both periodic and occurring aperiodic tasks subject to the model and the assumptions discussed in section II. In our experiments, we made a number of simplifying assumptions to allow for a more straightforward interpretation of the results: First, we assumed that all tasks consume the same average percentage utilization of the power being produced by the source.

We used a simulator which can generate aperiodic tasks with Poisson arrivals, exponentially distributed execution times, and uniformly distributed deadlines. The simulator implements an ED-H scheduler and our optimal admission controller. Figure 5 shows the percentage of tasks that make their deadlines (called success ratio) with and without acceptance test. The horizontal axis shows the generated incoming load which is varied from 10 percent to 90 percent of aperiodic load U_{ap} .

The success ratio at each load point is averaged over more than 1,000 aperiodic task arrivals, ensuring a sufficiently tight confidence interval in the results. Under low energy constraints, the normalized acceptance ratio of aperiodic tasks exhibits a notable variation with the periodic load U_p . At $U_p=0.1$, the ratio remains high, ranging from approximately 1 to 0.82, demonstrating that the admission control effectively exploits both slack time and slack energy, thereby outperforming policy that do not apply any acceptance test across all load levels. When U_p increases to 0.7, the ratio decreases to about 0.72, indicating a diminished capacity for accepting aperiodic tasks as the system approaches higher utilization. Importantly, the advantage of the admission test becomes more evident at higher loads: at low processing workloads, it guarantees acceptance of roughly 90% of aperiodic tasks; when the total processing load exceeds 50%, the acceptance ratio gradually declines but remains above 70%. In contrast, an admission-free approach experiences a sharper decline beyond 40% load,

dropping to approximately 58%. This trend highlights that task rejection is preferable to missed deadlines, and accepting a slight reduction in acceptance ratio is justified by the need to maintain deadline adherence. Overall, the results affirm that the admission control scheme maintains higher efficiency across varying load conditions, with no deadline misses observed under the control policy, whereas the non-controlled approach's success ratio deteriorates as load increases.

In Figure 6, the percentage of processor time used to execute jobs that successfully complete before their deadlines is presented, with the complement representing processor time spent either idling or executing jobs that miss their deadlines. The curves illustrate an expected increase in useful processing time as U_{ap} rises, regardless of whether an admission test is employed.

When analyzing the ratio of useful processing time relative to U_{ap} , notable differences emerge between the policies at various utilization levels. Interestingly, when varying U_{ap} from 0.1 to 0.3, both policies perform similarly, with minimal differences in their ratios. At a low utilization $U_p = 0.1$, the admission control policy shows the ratio increasing from approximately 13% to 71%, whereas the policy without admission control rises from about 13% to roughly 62%. This increase is attributed to the high rate of jobs meeting their deadlines under the admission test algorithm, which leverages slack time to improve job completion. Conversely, at a higher utilization $U_p = 0.7$, both policies experience declines, with the admission control policy rising from 48% to approximately 71%, and the non-admission policy from 48% to around 68%. As expected, the non-admission approach yields ratios roughly 30% lower than those of the admission test technique under full load. Although both approaches benefit from increased U_p , the admission control policy consistently outperforms the non-admission approach, especially at lower utilization levels, highlighting its effectiveness in optimizing useful processing time across different system loads.

V. RELATED WORKS

A. Scheduling hard deadline tasks with EH settings

[9] is the first study on real-time scheduling in monoprocessor energy-harvesting systems, restricted to the so-called frame based model. [10] introduce the Lazy Scheduling method (LSA) which is an idling scheduler based on the Earliest Deadline First rule. Optimality of LSA supposes that each job uses energy at a constant rate which is proportional to its execution time. HA-DVFS and EA-DVFS are two distinct DVFS-based algorithms that Liu et al. propose to expand LSA [11] [12]. When the processor is idle, energy saving is achieved by slow down. An optimal scheduler, called ED-H, is suggested by [13] with no restriction on the energy production and consumption models. As LSA, ED-H is based on the EDF rule. It defines, through a dynamic processor management procedure the periods when the processor should idle and when it should execute the highest priority task. [14] demonstrate how to optimize responsiveness of additional soft aperiodic tasks thanks to a slack stealing approach. [15]

suggest a real-time energy harvesting system that can manage the tasks with different performance levels. The idea is to cope with prediction inaccuracy. The resulting energy-resilient scheduling framework thus may adapt to changes in energy availability.

B. Online acceptance testing

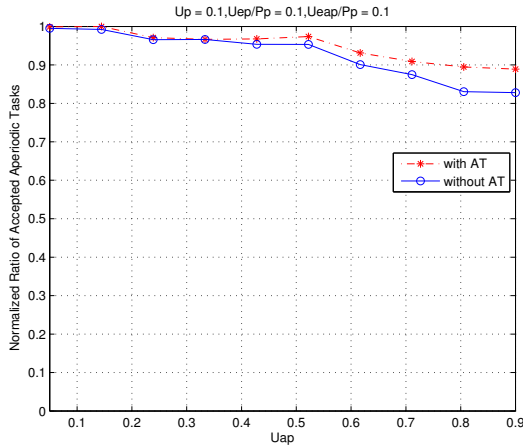
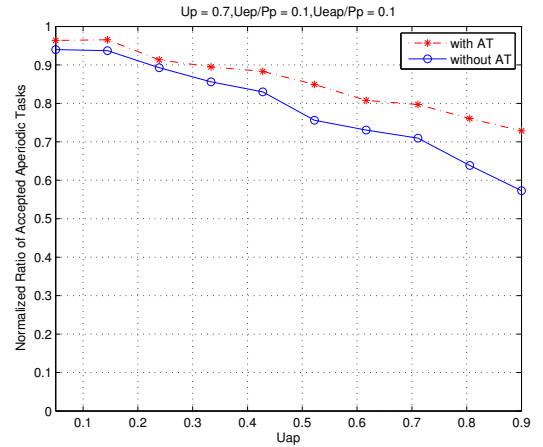
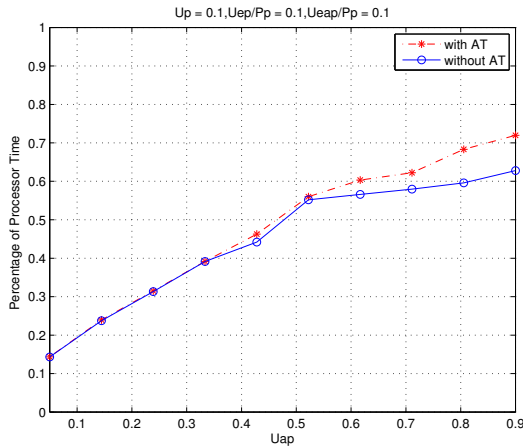
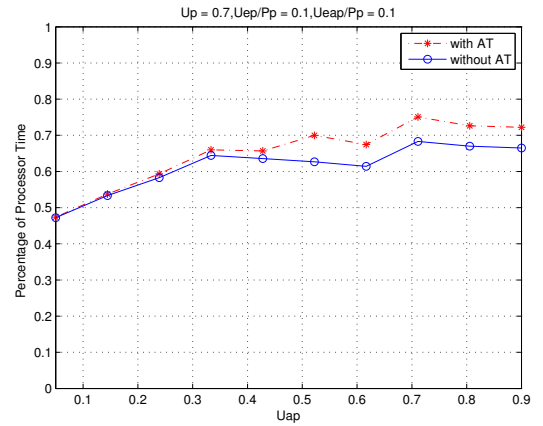
Determining whether incoming hard aperiodic tasks with unpredictable arrival times has been studied from about 40 years. Several kinds of acceptance tests have been proposed in the literature to decide whether adding a new hard deadline aperiodic task to a system will not cause other tasks to miss their deadlines [16] [17] [18] [19]. To our knowledge, no acceptance test in the literature takes into account variability in energy availability which characterizes self-powered devices.

VI. CONCLUSION

Self-powered IoT devices for wireless monitoring and control are being extensively used in many application domains such healthcare, smart cities, agriculture, industry, etc. A major issue is to make these systems as reliable as possible despite intermittent production of environmental energy used to supply them. We have proposed an efficient algorithm that makes online decisions on whether compute-intensive tasks such as IA tasks should be processed locally or offloaded to a more powerful computer resource provided by a fog server. Our final goal is to guarantee timing requirements for a maximum number of aperiodic tasks through effective integration of fog computing resources in the IoT infrastructure. This supposes to implement dedicated load balancing strategies over the diverse fog servers. Hence, the work presented in this paper is a small step towards this objective. The idea is to carefully consider both time and energy factors involved in computing and communication, employing energy-aware scheduling and offloading strategies, jointly.

REFERENCES

- [1] V. Lesch, M. Züfle, A. Bauer, L. Iffländer, C. Krupitzer, and S. Kounev, "A literature review of IoT and CPS—What they are, and what they are not," *J. Syst. Softw.*, vol. 200, no. C, June 2023. [Online]. Available: <https://doi.org/10.1016/j.jss.2023.111631>
- [2] M. Merenda, C. Porcaro, and D. Iero, "Edge Machine Learning for AI-Enabled IoT Devices: A Review," *Sensors*, vol. 20, no. 9, Art. no. 2533, 2020. [Online]. Available: <https://www.mdpi.com/1424-8220/20/9/2533>
- [3] J. Hornik, C. Ofir, and M. Rachamim, "Out of the fog: fog computing-enabled AI to support smart marketing management," *Management Review Quarterly*, vol. 75, no. 3, pp. 2543–2575, 2025.
- [4] F. Tsvetanov, "Integrating AI Technologies into Remote Monitoring Patient Systems," *Engineering Proceedings*, vol. 70, no. 1, p. 54, 2024.
- [5] K. K. Maguluri, "Artificial intelligence-driven patient monitoring: Real-time insights for proactive care," in *How Artificial Intelligence is Transforming Healthcare IT: Applications in Diagnostics, Treatment Planning, and Patient Monitoring*, Deep Science Publishing, 2025, pp. 50–71.
- [6] A. Rocha et al., "Edge AI for Internet of Medical Things: A literature review," *Computers and Electrical Engineering*, vol. 116, p. 109202, 2024.
- [7] P. Wägemann, T. Distler, T. Hönig, H. Janker, R. Kapitza, and W. Schröder-Preikschat, "Worst-Case Energy Consumption Analysis for Energy-Constrained Embedded Systems," in *Proc. 27th Euromicro Conf. Real-Time Systems*, 2015, pp. 105–114.

(a) for $U_p=0.1$.(b) for $U_p=0.7$.Fig. 5: Normalized success ratio with respect to U_{ap} .(a) for $U_p=0.1$.(b) for $U_p=0.7$.Fig. 6: Ratio of useful processing time with respect to U_{ap} .

- [8] H. Chetto and M. Chetto, "Some Results of the Earliest Deadline Scheduling Algorithm," *IEEE Trans. Softw. Eng.*, vol. 15, no. 10, pp. 1261–1270, Oct. 1989.
- [9] A. Allavena and D. Mosse, "Scheduling of Frame-Based Embedded Systems with Rechargeable Batteries," in *Workshop on Power Management for Real-Time and Embedded Systems (In Conjunction with RTAS 2001)*, 2001.
- [10] C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Real-time scheduling for energy harvesting sensor nodes," *Real-Time Systems*, vol. 37, no. 3, pp. 233–260, 2007.
- [11] J. Lu and Q. Qiu, "Scheduling and Mapping of Periodic Tasks on Multi-core Embedded Systems with Energy Harvesting," in *2011 Int. Green Computing Conf. and Workshops*, 2011, pp. 498–510.
- [12] S. Liu, S. Lu, J. Wu, and Q. Qiu, "Harvesting-Aware Power Management for Real-Time Systems with Renewable Energy," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, pp. 1–14, 2011.
- [13] M. Chetto, "Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 2, pp. 122–133, June 2014.
- [14] R. El Osta, M. Chetto, and H. El Ghor, "Optimal Slack Stealing Servicing for Real-Time Energy Harvesting Systems," *The Comput. J.*, vol. 63, no. 10, pp. 1537–1546, 2020.
- [15] M. Shirazi, L. Thiele, and M. Kargahi, "Energy-Resilient Real-Time Scheduling," *IEEE Trans. Comput.*, vol. 72, no. 1, pp. 69–81, Jan. 2023.
- [16] T. W. Kuo, L. P. Chang, Y. H. Liu, and K. J. Lin, "Efficient online schedulability tests for real-time systems," *IEEE Trans. Softw. Eng.*, vol. 29, no. 8, pp. 734–751, Aug. 2003.
- [17] E. Bini and G. C. Buttazzo, "Schedulability analysis of periodic fixed priority systems," *IEEE Trans. Comput.*, vol. 53, no. 11, pp. 1462–1473, Nov. 2004.
- [18] M. Silly, H. Chetto, and N. Elyounsi, "An optimal algorithm for guaranteeing sporadic tasks in hard real-time systems," in *Proc. 2nd IEEE Symp. Parallel and Distributed Processing*, 1990, pp. 578–585.
- [19] S. Ramos-Thuel and J. P. Lehoczky, "On-line scheduling of hard deadline aperiodic tasks in fixed priority systems," in *Real-Time Systems Symposium*, IEEE Computer Society Press, 1993.
- [20] Y. Li and J. Qiu, "Energy-aware scheduling for real-time systems," *IEEE Trans. Comput.*, vol. 60, no. 2, pp. 263–275, Feb. 2011.
- [21] X. Liu, Y. Li, and Z. Wang, "A survey on energy-efficient scheduling algorithms," *Journal of Systems and Software*, vol. 84, no. 7, pp. 1080–1090, 2011.