Advances and Challenges in Real-time Energy Harvesting Cyber Physical Systems

Mohammad El Ghor, Maryline Chetto, Hussein El Ghor

Advances and Challenges in Real-time Energy Harvesting Cyber Physical Systems

1st Mohammad El Ghor
Nantes Université, École Centrale Nantes
CNRS, LS2N, UMR 6004
F-44000 Nantes, France
mohammad.el-ghor@etu.univ-nantes.fr

2nd Maryline Chetto
Nantes Université, École Centrale Nantes
CNRS, LS2N, UMR 6004
F-44000 Nantes, France
maryline.chetto@ls2n.fr

3rd Hussein El Ghor
LENS Laboratory
Lebanese University
Saida, Lebanon
hussein.elghor@ul.edu.lb

***Abstract*—Designing Energy Harvesting Cyber Physical Systems (EH CPS) with real-time capabilities requires a sophisticated approach that integrates energy aware scheduling policies, predictive modeling, and dynamic task allocation. By effectively managing energy resources, these systems can adopt an energy neutral behavior: they ensure perpetual operation even in environments with intermittent energy availability and real-time requirements. This paper provides a brief overview of the state of the art about energy harvesting aware scheduling under real-time constraints. It describes the optimal scheduler called ED-H. Then, it highlights technical issues in the actual implementation of such a scheduler, particularly with regard to measurement of energy consumption and prediction of environmental energy production.**

***Index Terms*—Autonomous Cyber Physical System, real-time computing, energy harvesting, energy neutrality, Scheduling.**

I. Introduction

A. Energy Harvesting for Cyber Physical Systems

Cyber Physical Systems (CPS) and Internet of Things (IoT) are two interconnected technologies that play a crucial role in the development of modern smart systems [1]. They have applications in various domains including smart homes, healthcare, agriculture, transportation, industry and more. CPS integrate physical elements (like machines, sensors, and actuators) with computational and communication components. For example, through data analysis and machine learning, industrial CPS can predict when equipment might fail and schedule maintenance before a breakdown occurs. In the context of Smart Cities, CPS involve the interplay of sensors, actuators, communication networks, and computational systems to monitor and manage urban infrastructure. CPS monitor traffic flow, adjust traffic signals in real-time, and provide dynamic routing information to optimize traffic patterns.

CPSs should allow for real-time monitoring since monitoring and control require bounded responses to changes in the physical environment. Most of CPS are designed to operate on low power to maximize battery life. In applications where maintenance or battery replacement may be hazardous, Energy Harvesting (EH) provides a safer alternative [2] [3]. This technology refers to the process of collecting and storing energy from the environment to power electronic devices. By reducing reliance on external power sources and minimizing the need for battery replacements, energy harvesting promotes sustainability and can lead to cost savings over time. EH CPS i.e. the computer(s) of the CPS powered by energy harvesting can be deployed in any location where access for maintenance is difficult or costly. This flexibility in placement enhances the versatility of sensor applications [4]. Various renewable sources may be used, such as solar, thermal gradients, vibration, and ambient RF signals [5]. The choice of energy source depends on the specific environment. For example, in a factory, vibration based harvesting from machinery may be viable, while in an out-door setting, solar energy may be more practical. Energy harvested from the environment is converted from its native form into electrical energy using transducers. This energy is then stored in capacitors, batteries, or supercapacitors for later use so as the device survives when no energy can be harvested.

Work supported by e-Cobot company, Carquefou, France.

B. Real-time scheduling for EH CPS

Overall, energy harvesting plays a crucial role in enhancing the sustainability, autonomy, and operational efficiency of industrial CPS, contributing to the advancement of Industry 4.0 and smart manufacturing. However, EH CPS present a set of challenges that engineers and researchers need to address to create efficient and reliable systems. Facing these challenges requires a multidisciplinary approach, involving expertise in electrical engineering, computer engineering, mechanical engineering, and system integration. In particular, EH CPS should be provided with power management strategies in order to make them energy neutral i.e. to dynamically adjust the activity of the main component (processor), based on available energy levels. An energy-neutral system refers to a system where the amount of energy harvested from the environment is equal to or exceeds the amount of energy consumed by the system. In other words, the system is able to sustain its operation continuously without relying on an external power source or depleting an energy storage. Real-time scheduling in an EH CPS refers to the strategy adopted to control the execution of the different tasks on the computer of the EH CPS [6]. The goal is to ensure that the tasks execute within specified

time constraints expressed in terms of deadlines while also considering the intermittent nature of energy availability from the harvesting source. Note that these tasks are generally cyclic and their period is dependent on the physical process they control. Real-time scheduling, power management, and dimensioning are undoubtedly the difficult problems to be resolved. First, how can each task be given a priority based on its significance and/or urgency? Secondly, considering the nature of the energy source and the task timeliness requirements, how can the processing unit dynamically modify its activity to ensure its survival indefinitely? Thirdly, how to determine the dimensions of the energy storage unit and the harvester?

C. Contributions of the paper

The first objective of the paper is to bring to light the main issues in designing autonomous CPS with real-time requirements. The second objective is to report the state of the art and to describe the principles of ED-H, an optimal strategy for scheduling real time applications in EH CPS. We will also point out some unresolved challenges.
The rest of the sections in the paper is organized as follows. In Section 2, some background materials are given. In Section 3, a short review on scheduling techniques for EH CPS is carried out. Section 4 describes the details of the optimal energy harvesting aware scheduler, namely ED-H, for mono-processing EH CPS. It also includes an illustrative example. A discussion on applicability of this scheduler is presented in Section 5 followed by the conclusion in Section 6.

II. Background materials

A. Real-time scheduling with no energy limitation

The majority of real-time scheduling research has generally concentrated on models with processing requirements that are only expressed in terms of Worst Case Execution times (WCET) and where the tasks must be finished before a deadline. Numerous outcomes for both fixed-priority and dynamic-priority task scheduling have been attained, from the famous work of Liu and Layland five decades ago [7]. Jobs in periodic tasks can be statically and easily guaranteed in classical applications with no energy limitations using Rate-Monotonic (RM) priority assignment or Deadline Monotonic (DM). The RM algorithm uses periods to determine priority; the shorter a task's period, the higher the priority for all of its jobs. Priorities are assigned by the DM algorithm based on the relative deadline: the higher the priority, the shorter the relative deadline (which is assumed to be less than or equal to the period) [8]. In contrast, the foundation of EDF (Earliest Deadline First) is a dynamic task-level priority assignment and a fixed job-level priority assignment. The higher the priority, the earlier the job deadline. [7] proved optimality of EDF.

B. Energy harvesting aware scheduling

In energy autonomous CPS with intermittent energy supply, the previous classical real-time schedulers are not effective. This is because these schedulers are work-conserving i.e. the jobs are executed in ASAP mode, never making the processor idle if at least one job is pending for execution. We should be able to better understand the specific architecture of an effective energy harvesting aware scheduling algorithm by examining the potential consequences of job execution that involve both processing time consumption and energy consumption under strict timing requirements. Let us investigate the negative impact of energy variability in more detail. Using two jobs as an example (see figure 1), the upper and lower arrows stand for the jobs'respective release times and deadlines. The stored energy at any time t is denoted $E(t)$. A job, say J_1 that is currently running uses processing time and energy. As a consequence, no amount of energy remains to finish by the deadline, a newly occurring job, say J_2, regardless of its priority. Our instinct may tell us that when deciding whether to put the processor in sleep mode, we should take future job arrivals and their energy requirements into account. Example in figure 2 demonstrates this fact.

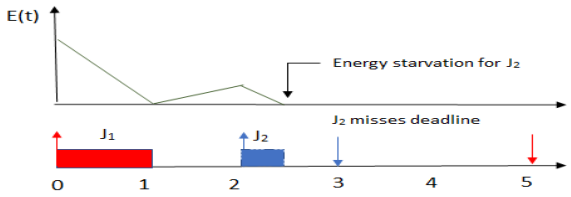

Fig. 1. Weakness of classical scheduling for energy harvesting systems.

So, we need to describe the jobs in terms of both processing time and energy consumption. Additionally, we need to accomplish the online prediction of environmental energy production in the near future as well as the online monitoring of the energy available in the storage unit to decide whether to make the processor busy v.s. idle (see figure 2). [9] demonstrate that the only schedulers that may be optimal are look-ahead-D schedulers, where D is the application's longest relative job deadline. Stated differently, any optimal scheduler that makes decisions at runtime needs to forecast the energy incoming and task arrival pattern over a minimum of D time units. Additionally, they demonstrate in [10] that EDF continues to be the best non-idling (also called work-conserving) scheduler even if not competitive under energy harvesting settings.

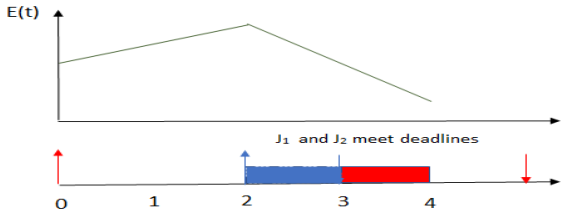

Fig. 2. Idling scheduling is necessary for EH CPS systems.

III. Related work

A. Scheduling for hard real-time applications

Let us assume applications where all the computing tasks have strict deadlines to meet. [11] firstly suggest a preemptive fixed-priority scheduling algorithm, known as PFP_{ASAP}, under the assumptions of a constant power supply and linearly energy-consuming tasks. Within the category of fixed priority assignment rules, the optimality of PFP_{ASAP} and schedulability condition were established. The first EDF-based algorithm to be proposed in the literature is the Lazy Scheduling Algorithm (LSA) [12]. LSA is optimal and may be used for periodic and/or non-periodic tasks with deadlines. Although the source power is represented as a time-varying variable, the rate at which energy is consumed by each job is constant. Using two distinct DVFS-based algorithms, HA-DVFS and EA-DVFS, Liu et al. expand LSA [13]. They use the idle time to slow down the processor, which helps them to save energy. [6] provides an optimal scheduler, called ED-H, with no restrictions regarding the task arrival profile and the energy source profile. It is shown how to incorporate additional aperiodic tasks so as to optimize responsiveness [14].

B. Best Effort scheduling for soft real-time applications

Let us assume applications where some computing tasks are authorized to miss deadlines. AsTAR, an energy-aware framework, has recently been developed to rank tasks according to importance [15]. By adjusting the computing service to the available environmental energy, it makes the optimal effort decisions. Recently, [16] examine a real-time energy harvesting system that supports periodic tasks with variable performance requirements. Because of prediction error, the suggested energy-resilient scheduling framework adapts to sudden variations in energy availability.

IV. Description of the optimal scheduler ED-H

A. Assumptions

We examine here an EH CPS that is composed of an energy harvester that uses a variable and predictable energy source, an energy storage device that can be a battery or super-capacitor as described in figure 3. The energy consumer is the single processor responsible for handling a set of deadline constrained tasks.

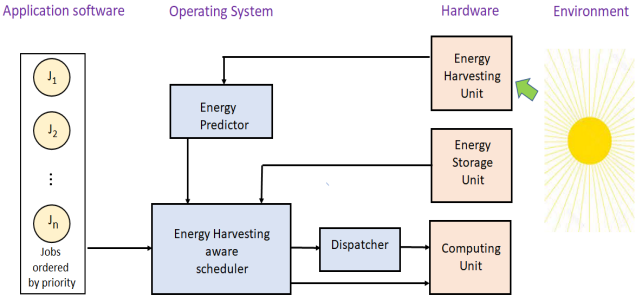

Fig. 3. Framework of an energy harvesting aware scheduler.

B. Main Principles

The ED-H scheduler is an extension of EDF with energy budgeting that controls timing and energy requirements as well as the rate of storage unit replenishment. ED-H's central idea is to dynamically adjust the processor activity to runtime harvested power fluctuations while respecting deadlines, by calculating the so-called slack time and preemption slack energy.

The *slack time* at current time t is the length of the longest interval starting at t during which the processor can stay idle without leading to deadline violations. The *preemption slack energy* at current time t is the maximum energy that can be consumed by the highest priority job ready for execution at time t, without provoking energy starvation in the future. In other terms, if the preemption slack energy is zero at time t, the processor has to immediately enter the idle state. For a more detailed explanation of the slack time and preemption slack energy, see [6]. For example, figure 4 shows that the preemption slack energy should be computed at time 10: a job of task τ_2 with deadline 18 releases whereas a higher priority job of task τ_1 with deadline 17 will release at time 12.

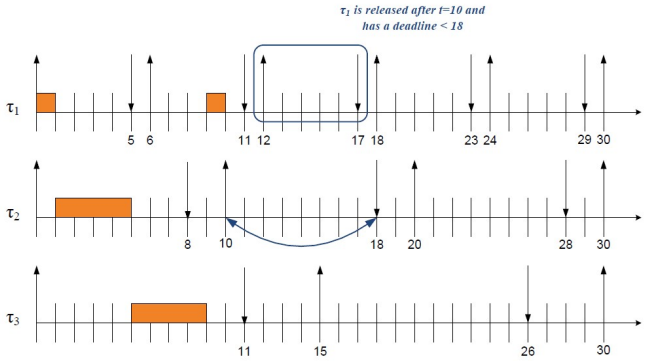

Fig. 4. Illustration of the slack energy concept

Energy availability, preemption slack energy and slack time is respectively denoted by $E(t)$, $PSE(t)$ and $ST(t)$ in the pseudo-code of ED-H:

```
1:  while (1) do
2:    while PENDING=true do
3:      while (E(t) > E_min and PSE(t) > 0) do
4:        execute()
5:      end while
6:      while (E(t) < E_max and ST(t) > 0) do
7:        wait()
8:      end while
9:    end while
10:   while PENDING=false do
11:     wait()
12:   end while
13: end while
```

PENDING is a boolean which equals true whenever there is at least one job in the ready task list. Function $wait()$ puts the processor in sleep mode and function $execute()$ puts the

processor in active mode. Jobs are ordered according to their urgency. We notice that we do not allow jobs to run after E_{min}. We start charging the energy storage when, either it is empty ($E(t) = E_{min}$) or there is no sufficient energy to guarantee the feasible execution of the current and all future occurring jobs i.e. the system has no more preemption slack energy (line 3). The processor is let inactive until the time instant when either the energy storage unit has completely replenished or there is no more slack time (line 6). ED-H checks if there are ready jobs to be executed. If not, the processor is made idle until the next release of job (line 10). Let us note that, whenever the processor does not execute any job, the slack time linearly decreases with time. ED-H tests whether $E(t) > E_{min}$ where E_{min} is a threshold for the level of the energy storage unit. This threshold can be defined as the maximum energy consumed by any job.

C. Illustrative example

Consider a periodic task set Γ that is composed of three periodic tasks, $\Gamma = \{\tau_i \mid 1 \leq i \leq 3\}$ and $\tau_i = (C_i, D_i, T_i, E_i)$ where C_i, D_i, T_i and E_i respectively gives the computation time, the deadline, the period and the energy consumption of task τ_i. Let $\tau_1 = (1, 5, 6, 12)$, $\tau_2 = (2, 8, 10, 15)$ and $\tau_3 = (4, 11, 15, 22)$. We assume that the energy storage capacity is $E = 25$ energy units at $t = 0$. For simplicity, we assume that $E_{min} = 0$, $E_{max} = E$ and the rechargeable power is constant and equal to 5 ($P_r = 5$). Moreover, we assume that the maximum consumption power is 10 energy units.
Let us comment the ED-H schedule produced between time 0 and time 30.

- At time $t = 0$, $E(0) = 25$. τ_1, as the highest priority job, runs and finishes at $t = 1$. $E(1) = E(0) - E_1 + (P_s \times C_1) = 25 - 12 + 5 = 18$ energy units. As there is no job released after 0 with deadline less than that of τ_1, the slack energy does not require to be computed at $t = 0$. The energy level in the storage unit permits to satisfy the energy requirement of τ_1.
- At time $t = 1$, τ_2 is the highest priority job ready to be processed. τ_2 finishes at $t = 3$ where $E(3) = 13$ energy units.
- At time $t = 3$, τ_3 is the highest priority job ready to be processed. τ_3 finishes at $t = 7$ where $E(7) = 11$ energy units.
- At time $t = 7$, τ_1 is the highest priority job ready to be processed. τ_1 finishes at $t = 8$ where $E(8) = 4$ energy units.
- From time $t = 8$ up to $t = 10$, the processor remains idle because there are no pending job. During that time interval, the energy storage will recharge and the energy level at $t = 10$ is given by $E(10) = 14$ energy units.
- At $t = 10$, τ_2 has the highest priority task ready to be processed. Here, computation of the preemption slack energy is necessary because τ_1 with deadline less than deadline of τ_2 will be released after time 10 (figure 4). The preemption slack energy of the system at 10 is 27 since the slack energy of τ_1 at 10 given by $E(10) + \int_{10}^{d_1} E(t)dt - E_1$ is 37 and the slack energy of τ_2 at 10 given by $E(10) + \int_{10}^{d_2} E(t)dt - (E_1 + E_2)$ equals 27. τ_2 is autorized to execute immediately.
- At time $t = 12$, $E(12) = 9$. τ_1 has the highest priority with no future job having a lower deadline. Consequently no slack energy computation is required. It is executed till $t = 13$, where $E(13) = 2$ energy units.
- From $t = 13$ up to $t = 15$, the processor is idle because there are no ready job. Consequently the battery will recharge leading to $E(15) = 12$ energy units.
- At $t = 15$, τ_3 is ready and has the highest priority. Preemption slack energy requires to be computed because τ_1 more urgent than τ_3 will be released after time 15. The preemption slack energy is 33 energy units. τ_3 is executed till $t = 18$, where preemption occurs since τ_1 has a higher priority than τ_3.
- At, $t = 18$, we can evaluate the maximum energy that remains to be consumed by τ_3 as follows: we memorize the energy level of the storage unit at every scheduling point i.e. whenever a preemption occurs or a task begins execution. At $t = 18$, τ_1 is the highest priority task ready to be processed. But there is no sufficient energy in the battery for execution. So, the processor is made inactive as long as the energy storage has not filled completely ($E = E_{max}$) and the latest start time of the jobs has not been attained i.e. at $t = 22$.
- At $t = 22$, $E(22) = 25$ energy units. Now, τ_1 has the highest priority and it is executed till $t = 23$, where $E(23) = 18$ energy units.
- At $t = 23$, τ_3 completes its execution till $t = 24$ where $E(t) = 8$ energy units.
- At time $t = 24$, τ_2 has the highest priority. τ_2 is executed up to $t = 26$, where $E(26) = 3$ energy units.
- At time $t = 26$ τ_1 is ready and has the highest priority, but there is no sufficient energy in the battery for execution. So, the processor enters the idle state and the energy storage recharges till $t = 28$ where $E(28) = 13$ energy units.
- At time $t = 28$, τ_1 is executed up to $t = 29$, where $E(29) = 6$ energy units.
- Finally, the processor remains idle from $t = 29$ to $t = 30$ where $E(30) = 11$ energy units.

The final Gantt chart that represents the ED-H schedule on Γ is illustrated by figure 5.

D. Simulation results

We report here part of an experiment that permits to show the gain obtained by ED-H regarding the deadline miss rate. For a given energy consumption ratio, we repeat the simulation for 1000 periodic task sets. We give here the results for 0.6 as processing utilization ratio. At the beginning of the simulation, the energy storage is full. Figure 6 corroborates the outperformance of ED-H on other schedulers, notably in terms of deadline miss rate. Let us define EH-EDF, an idling variant of EDF that is simple to implement and doesn't require energy harvesting prediction in contrast to ED-H. The idea underlying

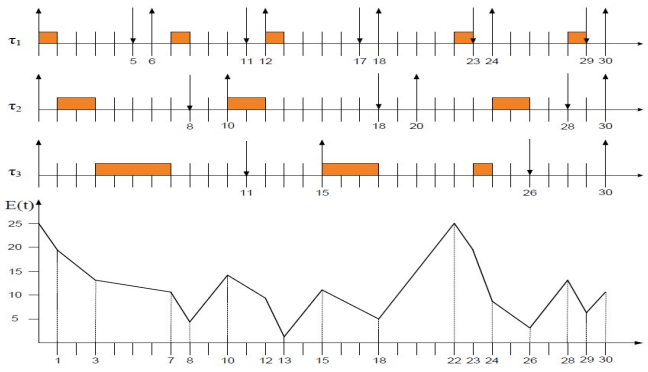

Fig. 5. The ED-H schedule.

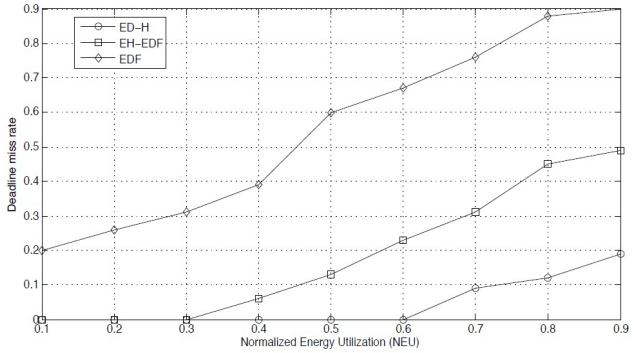

Fig. 6. Outperformance of ED-H on other schedulers.

EH-EDF is to schedule jobs in accordance to EDF as soon as possible i.e. consuming energy in a greedy manner. The processor enters the idle state and jobs are postponed as much as possible whenever the energy storage is deplenished. Until the storage completely replenishes or the slack time drops to zero, the scheduler allows the processor to stay inactive. The benefit of the EH-EDF policy is be operational even under unpredictable future energy incoming.

Figure 6 shows the deadline miss rates with the Normalized Energy Utilization (NEU) sweeping from 0.1 to 0.9, while processor utilization and capacity of the energy storage are fixed parameters. We define NEU as the ratio of mean average consumption power per average production power. It is demonstrated that, in comparison to EDF and EH-EDF, ED-H achieves significantly lower deadline miss rates in all energy utilization settings. For example, when NEU rises to 0.6, EDF and EH-EDF record deadline miss rates roughly equal to 60% and 20%, respectively while all deadlines are met under ED-H (see [17] for additional simulation results).

V. Implementation challenges

A. Assessment of energy consumption

Effective operation of an EH CPS requires to address additional technical challenges compared to a traditional battery-powered system. The scheduling and processor management framework described previously has been proved optimal under simplistic assumptions: precise assessment of energy consumed by the wireless device and precise prediction of energy produced by the environmental source. Estimating the worst-case energy usage of any task can be straightforward when confined to the impact of the processor. Nonetheless, energy consumption may be contingent upon the work completed by other energy consumers, including memory, timers, WiFi transceivers, LEDs, etc.

In most of studies, processor energy consumption in the idle state was deemed insignificant. In fact, when a processor is idle, it uses energy that is contingent upon a number of parameters, such as the processor's architecture and the amount of power used by components that remain active during idle times. An efficient method to consider this phenomenon is to keep a security stock of energy in the energy storage so as to guarantee the system's survival regardless of the processor's state. In a traditional battery-powered CPS, the precise amount of energy in the storage unit that is available for use is known in advance. In an EH CPS, the actual stored energy may not be as predicted. Researching particular methods that try to reduce the rate of missed deadlines in circumstances where the harvested energy is overestimated could be one line of inquiry for future work.

B. Operating system facilities for energy management

The real-time operating system (RTOS) is a fundamental component of the CPS. It provides the necessary software infrastructure to manage computation tasks, resources, and communication within the system, while meeting the timing constraints (deadlines) imposed by the physical world. Choosing the right RTOS and configuring it appropriately is a critical aspect of CPS development. It is important to consider the specific requirements of the hardware platform, the nature of the energy source, and the overall power management strategy. The RTOS has to obtain accurate and timely information so as to trigger energy-related events. To our knowledge, energy harvesting support is not part of the core of any RTOS. Note that the scheduler ED-H, recently, has been implemented in the Xenomai kernel which operates alongside a general-purpose operating system, usually Linux [18].

On line monitoring of the state of the energy storage serves to notify the OS when a specified threshold is reached. Such functionality is essential as it enables to adjust dynamic power management according to the amount of energy actually available. In addition, the system should be equipped with an energy prediction mechanism to avoid any future energy starvation. In ED-H, the so called preemption slack energy represents the highest amount of energy that could be continuously consumed by any job in execution. As a consequence, its value needs to be estimated at least before starting the execution of every job and possibly at regular time instants during execution. This implies to call for the energy predictor [19]. If the preemption slack energy falls below a certain threshold, the OS should be notified. And the device should switch in the sleep mode so as the energy storage unit may recharge. At this instant just before the system enters the

sleep mode, sufficient energy should be left to save the system state in non volatile memory. Identically, the recharging phase terminates as soon as there is no more slack time or enough energy becomes available in the storage unit.

C. Prediction of energy production

Predicting ambient energy availability is indeed a challenging task due to the dynamic and unpredictable nature of energy sources such as solar, wind, or vibration. Classical prediction techniques use historical energy harvesting data to forecast future energy availability [20] [21]. The moving average technique, for instance, assigns equal weight to all historical data and forecasts future values based on the averages of previous observations. Different weight factors for historical values that vary according to the distance from the current time are used by the exponential smoothing technique. Reliability issues also arise from the variability and uncertainty in these energy sources. Here are some reasons why prediction algorithms for ambient energy may face challenges: Firstly, ambient energy sources are highly influenced by environmental factors, and their availability can vary significantly over time. Changes in weather conditions, seasons, or natural events can affect energy generation. Secondly, the environment may introduce noise or disturbances that impact the accuracy of sensors or data used by prediction algorithms. In addition, sensors may have limited precision, leading to inaccuracies in the data used for predictions. To address these challenges, researchers are exploring various strategies [22] [23]. This includes data Fusion which consists in combining data from multiple sources so as to enhance the robustness of predictions. Machine Learning and AI can also learn patterns from historical data and adapt to changing conditions. Nonetheless, predicting ambient energy remains a complex task, and achieving high reliability requires ongoing research and development. Failure in correctly predicting harvested energy could have serious consequences since it may involve unanticipated depletion of the energy storage. As a consequence, it will be necessary to provide any autonomous device with resilience capabilities so that it may survive even in a degraded mode, in the face of energy starvation [16].

VI. Conclusion

Self-powered wireless monitoring and control systems will be widely deployed in many application fields including industry, smart cities, healthcare, etc. One major challenge is to make these systems as stable as possible despite intermittent ambient energy used to supply them. The two difficult characteristics that set them apart from battery-operated computing systems are timeliness and energy neutrality. Energy harvesting rate prediction is clearly a central technological barrier to overcome. Additionally, we suggest to conduct research on implementing the optimal on-line energy aware scheduler EDH jointly to a suitable recovery mechanism for guaranteeing reliability and survivability of the EH CPS in any circumstance [24].

References

[1] C. Lu et al. (2016) Real-Time Wireless Sensor-Actuator Networks for Industrial Cyber-Physical Systems. In Proceedings of the IEEE, 104(5), 1013–1024, doi: 10.1109/JPROC.2015.2497161

[2] Yildiz, F. (2009) Potential ambient energy harvesting sources and techniques. The Journal of Technology Studies. 35(1), 40–48.

[3] Chalasani, S.; Conrad, J. M. (2008) A survey of energy harvesting sources for embedded systems. In Proceedings of the IEEE Southeastcon 2008, Huntsville, AL, USA, 442–447.

[4] Ma D, Lan G, Hassan M., Hu W, Das S. K. (2020) Sensing, Computing, and Communications for Energy Harvesting IoTs: A Survey. IEEE Communications Surveys and Tutorials 22(2), 1222–1250.

[5] Newell, D, Duffy M. (2019) Review of Power Conversion and Energy Management for Low-Power, Low-Voltage Energy Harvesting Powered Wireless Sensors. IEEE Transactions on Power Electronics, 34(10), 9794–9805.

[6] Chetto M. (2014) Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems. IEEE Trans. on Emerging Topics in Computing, 2(2), 122-133.

[7] Liu C.l., Layland J.w. (1973) Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the Association for Computing Machinery, 20(1), 46–61.

[8] Davis R., Cucu-Grosjean L., Bertogn, M., Burns, A. (2016) A review of priority assignment in real-time systems. Journal of systems architecture, 65, 64–82.

[9] Chetto M., Queudet, (2014) A. Clairvoyance and Online Scheduling in Real-Time Energy Harvesting Systems. Real-Time Systems Journal, 50(2), 179–184.

[10] Chetto M., Queudet, A. (2014) A Note on EDF Scheduling for Real-Time Energy Harvesting Systems. IEEE Transactions on Computers,63(4), 1037–1040.

[11] Abdeddaim Y. Chandarli Y., Masson, D. (2013) The Optimality of PFPASAP Algorithm for Fixed-Priority Energy-Harvesting Real-Time Systems. In Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS).

[12] Moser C., Brunelli D., Thiele, L., Benini L. (2007) Real-time scheduling for energy harvesting sensor nodes. Real-Time Systems, 37(3), 233–260.

[13] Lu J., Qiu Q. (2011) Scheduling and Mapping of Periodic Tasks on Multi-core Embedded Systems with Energy Harvesting. In International Green Computing Conference and Workshops, pp. 498–510

[14] El Osta R., Chetto M.; El Ghor H. (2020) Optimal Slack Stealing Servicing for Real-Time Energy Harvesting Systems. The Computer Journal, 63(10), 1537–1546.

[15] Yang F., Thangarajan A. S., Ramachandran G. S. and Joosen W. (2022) AsTAR: Sustainable Energy Harvesting for the Internet of Things through Adaptive Task Scheduling. ACM Transactions on Sensor Networks, 18(1).

[16] Shirazi M., Thiele L., Kargahi, M. (2023) Energy-Resilient Real-Time Scheduling. IEEE Transactions on Computers, (72,1), 69–81.

[17] Chetto M., El Ghor H. (2019) Scheduling and power management in energy harvesting computing systems with real-time constraints. Journal of Systems Architecture, 98, 243–248 .

[18] Mohamed Abdulla M.I. (2023) Real-time and energy constrained cobotic systems, PhD Thesis, University of Nantes, France.

[19] Cammarano A., Petrioli C., Spenza D. (2016) Online Energy Harvesting Prediction in Environmentally Powered Wireless Sensor Networks. IEEE Sensors Journal, 16(17), 6793–6804.

[20] Kansal A., Hsu J., Zahedi S. and Srivastava M.B. (2007) Power Management in Energy Harvesting Sensor Networks. ACM Transactions on Embedded Computing Systems, 6(4).

[21] Bergonzini C. , Brunelli D. and Benini L. (2010). Comparison of energy intake prediction algorithms for systems powered by photovoltaic harvesters. Microelectronics Journal, 41, (11), 766–777.

[22] Balaji S. and Karthik S. (2023) Comparative Study of Various Machine Learning and Deep Learning Techniques for Energy Prediction and Consumption Using IoT Modules. Proceedings of the International Conference on Cognitive and Intelligent Computing, 99–105.

[23] Muhammad Qureshi H.K., Saleem U., Saleem M., Pitsillides A. and Lestas M. (2017) Harvested energy prediction schemes for wireless sensor networks: Performance evaluation and enhancements. Wirel. Commun. Mob. Comput., 6928325.

[24] Chetto M. and El Osta R. (2023) How to provide fault-tolerance capabilities to energy-autonomous sensor systems with real-time constraints? Proceedings of the 22nd IFAC World Congress, Yokohama, Japan.