Energy-Adaptive Real-time Sensing for Batteryless Devices

Mohsen Karimi, Yidi Wang, and Hyoseung Kim
University of California, Riverside
mkari007@ucr.edu, ywang665@ucr.edu, hyoseung@ucr.edu

***Abstract*—The use of batteryless energy harvesting devices has been recognized as a promising solution for their low maintenance requirements and ability to work in harsh environments. However, these devices have to harvest energy from ambient energy sources and execute real-time sensing tasks periodically while satisfying data freshness constraints, which is especially challenging as the energy sources are often unreliable and intermittent. In this paper, we develop an energy-adaptive real-time sensing framework for batteryless devices. This framework includes a lightweight machine learning-based energy predictor that is capable of running on microcontroller devices and predicting the energy availability and intensity based on energy traces. Using this, the framework adapts the schedule of real-time tasks by effectively taking into account the predicted energy supply and the resulting age of information of each task, in order to achieve continuous sensing operations and satisfy given data freshness requirements. We discuss various design choices for adaptive scheduling and evaluate their performance in the context of batteryless devices. Experimental results show that the proposed adaptive real-time approach outperforms the recent methods based on static and reactive approaches, in both energy utilization and data freshness.**

I. Introduction

Latest batteryless embedded devices are designed to harvest energy from the environment energy resources, i.e., Radio Frequency (RF) signals, body movements, solar energy, wind, etc., and to provide usable data to the end user despite intermittent energy supplies. The variety of their applications such as health monitoring and smart agriculture is due to the fact that these devices are more reliable, have longer lifetime, require very few maintenance, and can be implemented in extreme environments where batteries cannot operate, e.g., high temperature, or for high pulse/high drain applications.

Although many batteryless sensors are being developed these days, conducting reliable task execution on batteryless devices is still challenging especially when real-time sensing tasks, that collect data from sensors, are considered. Energy sources for harvesting devices are usually unreliable and intermittent while real-time systems necessitate periodic/sporadic task executions with guaranteed deadline or bounded deadline misses for hard or soft real-time requirements respectively.

Furthermore, the amount of energy that can be stored in this type of devices is usually small due to the small size of energy storage capacitors that can be used [1]–[5]. Therefore, any long uninterruptible task execution without a proper power management is difficult if not impossible. This calls for a precise energy predictor as well as an energy-aware task scheduler to properly adapt task executions based on the energy availability while satisfying the entire system's real-time requirements.

In many sensing applications, even though sensing tasks are periodic, maintaining a desired level of data freshness of tasks is often more important than increasing the number of executed jobs or reducing the number deadline misses. For example, for health monitoring systems, the data freshness of each sensor is more critical than the total number of samples from all sensors during a fixed time interval. Data freshness is also known to be a critical criteria in databases [6], [7]. Existing methods for task scheduling on batteryless devices can be categorized into *static* and *reactive* approaches. The static approaches [4], [8] target periodic task execution with known charging behavior. They aim to meet the deadlines of a given taskset when the system is provided with an assumed amount of energy supply. The reactive approaches [9], [10] consider event-driven tasks and aim to minimize job response times in a best effort manner whenever the system has energy to run. However, neither of these approaches are suitable for real-time sensing applications where ensuring consistent data freshness over a long period of time under changing environmental conditions is more important than focusing on individual job-level deadlines or response times.

This paper presents an energy-adaptive real-time sensing framework for batteryless devices. We consider Age of Information (AoI) as the metric to measure data freshness of each task. The proposed framework consists of a machine learning-based energy predictor unit, to estimate the energy reception rate of the device for a future time intervals based on the previous energy samples, as well as a task scheduler unit that utilizes the predicted energy and provides an efficient schedule for real-time sensing tasks while satisfying the system data freshness requirements. In this paper, we particularly focus on solar energy prediction, but our proposed method with minor modifications can be applied to any energy resource that have a pseudo-periodic behavior at different time scales.

The rest of the paper is structured as follows: Section II reviews previous work. In Section III, some backgrounds and the system model are presented. The proposed method is described in Section IV. Evaluation results are presented in Section V. Finally, Section VI concludes the paper.

II. Related Work

There are many works that have focused on predicting the energy received by energy harvesting devices from various power sources. In this paper, we mainly focus on the prediction of energy received from solar radiation. In [11], the exponentially weighted moving average (EWMA) is proposed to predict the energy of each time slot based on the moving average of the same time slot at previous days. The method performs well in stable weather conditions; however, the error is substantially high for day-to-day weather changes. In [12], the authors proposed the weather-conditioned moving average (WCMA) method based on EWMA and provided an adaptation factor which considers the previous time slots on the same day to mitigate the error for day-to-day weather changes. Pro-Energy [13] provides an energy predictor by defining different profiles for the prediction. For each time slot, the predictor chooses one of the profiles based on the similarity of the previous data to each of the profiles and then makes prediction based on a similar method as EWMA. Afterwards, it checks if the current data can be stored as a new profile or be substituted with another profile in the profile list. Despite its relatively high accuracy in smaller time frames, the method requires a large amount of memory and computation than EWMA and WCMA to store, compare, and update the profiles, which leads to a substantial overhead to task execution on batteryless embedded devices.

An extensive amount of work has been done in solar radiation prediction. Most existing approaches either take into account other sensory parameters such as temperature, pressure, humidity, and wind speed [14]–[17] which are not always available, or employ complex models that can lead to a computation overhead of several seconds and an extensive amount of memory, making them unusable for batteryless devices with limited capabilities [18]. In [19], the authors provide a hybrid model based on Long Short Term Memory (LSTM) network and empirical mode decomposition (EMD). Although the provided method has better performance than WCMA method in some of the cases, in the experiment done by the authors, the average of Mean absolute percentage error (MAPE) of prediction for both WCMA and EWMA are still better. Furthermore, [20] provides a Q-learning based solar energy predictor for wireless sensor networks. The author's experiments show that their proposed method outperforms EWMA in all the cases and Pro-Energy in some of the cases. However, Pro-Energy performs better with some weighting factor parameters. For multiple battery-powered energy harvesting devices connected to a single base station, [21] provides a two layer reinforcement learning network to predict the battery state and control the access of each joint in order to minimize the prediction loss and maximize the transmission sum rate, respectively. The authors' simulation results show that their method has better performance than myopic, round-robin, and random scheduling policies.

Regarding task execution on batteryless devices, there are many methods that focus on scheduling tasks on an intermittently-powered devices while maintaining the memory consistency of the device and the ability to resume task execution in the presence of power failures. The scheduling papers can be categorized into static and reactive approaches. For static approaches, the authors of [8] provide online and offline schedulers which ensure the schedulability of taskset for a given energy harvesting rate at any time as a priori. Although they could provide a reasonable schedule given the charging rate, the schedulability overhead is significant. They also considered tasks to be completely preemptive which makes it incompatible with sensing tasks as well as the state-of-the-art programming models and kernels designed for batteryless devices. [4] also takes a static approach and it provides a real-time task scheduler that considers the energy harvesting device as a periodic energy resource. They provided analysis for hard real-time tasks under the Rate Monotonic (RM) and the Earliest Deadline First (EDF) scheduling policies and demonstrated a hardware implementation to assess the performance of the scheduler. Although the method is shown to perform well when a steady minimum energy provision rate is provided and can tolerate a certain duration of missing energy sources, in real-world energy harvesting scenarios with weather conditioned energy sources like solar energy, it may underutilize surplus energy or fail to follow a given schedule, resulting in deteriorated data freshness.

Among reactive scheduling methods, [9] provides a event-based kernel developed for timely execution of tasks on intermittently powered devices. It schedules the tasks whenever the energy is available without considering the energy rate and conserving energy for the future and therefore, unable to prevent any deadline misses. In [10], the authors provide a low overhead scheduling method to degrades the performance of the scheduler in case of low energy harvesting rates to prevent deadline misses. However, the harvested power is modeled as Gaussian distribution which makes it unable to be adaptable to many of the real-world applications.

III. System Model

In this section, we provide a background of the concepts used in the paper as well as the properties of the system used in the rest of the paper.

A. Age of Information

We use the Age of Information (AoI) to quantify the data freshness of each task. We define the AoI of a task τ_i at time t, $A_i(t)$, as the time elapsed since the latest output of the task was generated. Fig. 1 shows the AoI of a task τ_1 in a taskset with 3 tasks. For example, after first job execution of task τ_1 at time t_1, the output starts to age and the AoI of τ_1 increases linearly with time until the second job of τ_1 finishes its execution. At time t_2 a new fresh output is generated and AoI of τ_1 changes from $t_2 - t_1$ to 0. To find the average AoI we need to consider the area below the graph shown in Fig. 1 or calculate the following equation.

$$\mu_{A_1} = \frac{\int_{t_1}^{t_4} A_1(t)\,dt}{t_4 - t_1} = \frac{\sum_{i=1}^{3}(t_{i+1} - t_i)^2}{2 \times (t_4 - t_1)} \tag{1}$$

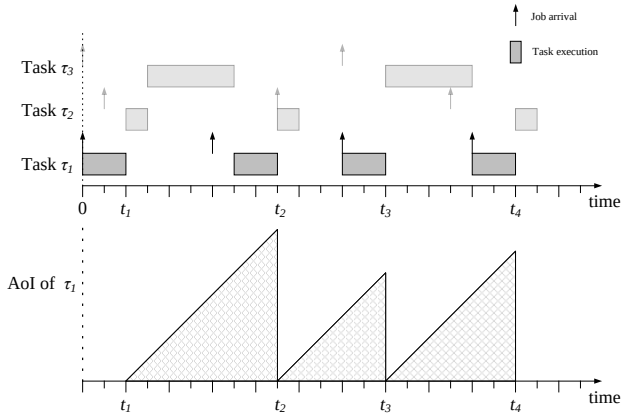

Fig. 1. Age of Information of task τ_1

where $A_1(t)$ is the AoI of the task τ_1 at time t, μ_{A_1} is the mean of the AoI of τ_1 between time t_1 to t_4. It should be noted that we calculate the AoI after the first job execution, i.e., after t_1 for task τ_1 in the example, because there is no output available before t_1.

B. Task Model

In this paper, we consider AoI as the metric to measure the data freshness of a task. We characterize a task τ_i in a taskset Γ by $\tau_i = (C_i, T_i, D_i, MTA_i)$, where C_i, T_i, D_i, and MTA_i are the worst-case execution time, period, deadline, and maximum tolerable AoI of the task i, respectively. It should be noted that in most of the applications the provided MTA is larger than and D_i is smaller than the period of the tasks, i.e. $\forall i \leq n \mid MTA_i \geq T_i \wedge D_i \leq T_i$. Most of the state-of-the-art kernels and programming models designed for intermittently powered devices [2], [9], [22], [23] consider non-preemptive tasks to ensure forward progress and memory consistency in case of power losses. Furthermore, in sensing applications, any power loss during the sensing would lead to a failure in sensing operation and sensor data could not be obtained at all. Therefore, we consider all the sensing tasks to be non-preemptable. We also utilize the charging model that is proposed in [4] which considers m_{P_i} as the discharging rate for task τ_i. Based on [4], considering the charging rate of the system to be fixed as m_a and the discharging rate of task τ_i to be m_{P_i}, the charging time required for task τ_i can be calculated as

$$Q_i = \frac{(m_{Pi} - m_a) \times C_i}{m_a} \tag{2}$$

where Q_i is the charging time required before starting the execution of task τ_i to guarantee its completion without any interrupt, i.e., the system needs to be charged for at least Q_i time units with the charging rate of m_a before task τ_i with the discharging rate of m_{P_i} starts its execution. Otherwise, the device's energy would deplete in the middle of execution of τ_i and the task's execution would fail. It should be noted that the charging time Q_i can be performed at any time before starting τ_i and is not required to happen in a single or continuous charging session.

C. Energy Harvesting

Batteryless devices use an energy harvester unit to convert energy from the incoming energy source to a type of energy, usually electricity, that can be stored in energy storage unit. The energy storage unit usually contains capacitors to store the energy so that it can be later used by the device when it is needed. They also usually contain an energy monitoring unit which measures the energy level of the energy storage unit at fixed time intervals and can estimate the average harvesting rate of the past interval from the measurement, e.g., by calculating the difference of the energy at the beginning and the end of the time interval divided by the time interval length. The charging rate of the system is considered fixed during each prediction time interval, T_{adapt}, which we will discuss later in the paper.

IV. Proposed Framework

The proposed framework consists of five units as shown in Fig. 2. As discussed earlier, the energy harvester converts energy to electricity, the harvested energy is stored in an energy storage unit (e.g., capacitors), and the energy monitoring unit contains the circuitry to measure the energy harvesting rate. As discussed in the previous section, the actual energy harvesting rate of each time interval can only be calculated at the end of that time interval.

The harvesting rate computed by the energy monitoring unit is fed in the energy predictor unit to compute a predicted energy rate for the next time slot. Finally, the predicted energy rate from the previous state is used by the task scheduler unit to efficiently schedule real-time tasks. The prediction is triggered by the task scheduler and occurs periodically based on the given time interval, T_{adapt}, which is configurable by the user (it is set to 30 minutes in our experiment). At the beginning of each time slot, the scheduler updates the schedule according to the predicted energy harvesting rate for the next time slot. In the remaining parts of this section, we further provide a more detailed explanation of the designed energy predictor as well as the proposed task scheduler unit.

A. Energy Predictor

Due to the limited processing capability of batteryless devices, we need to use an energy prediction model that (1) does not require substantial processing and (2) produces relatively accurate prediction. Although the existing methods [11]–[13] perform reasonably for consecutive days with similar exposure to the sun, they cannot properly predict fluctuations during the day, e.g. when clouds cover the sun temporarily or when there is a cloudy day after a sunny day. Fig. 3 shows an example of power reception from solar panels at May 6 and May 7, 2016 used from the NREL dataset [24] as well as the predicted values by EWMA and WCMA methods (the predictions are conducted every 15 minutes). As it is shown in the Fig. 3 both methods failed to adapt the prediction. This behavior motivated us to design a machine learning based predictor. We design a Neural Network (NN) predictor to predict the solar energy rate received from solar panels.

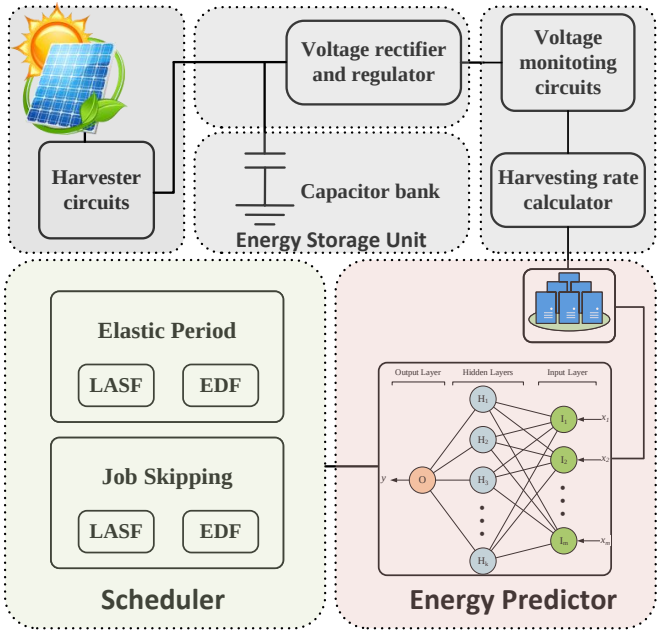

Fig. 2. Overview of the proposed energy-predictive scheduler

1) Neural Network Based Predictor: We present a light-weight model architecture to make it amenable to run on batteryless devices that mostly use microcontrollers. This is the simplest yet feasible model we could come up with, without losing opportunities to learn dependencies in time-series input. We divide the time into 30 minutes of time slots, i.e., $T_{adapt} = 30min$, since solar radiation does not drastically change during this amount of time. Furthermore, it has been used as a standard prediction duration by many of the previous studies such as Pro-Energy and WCMA. We designed a neural network that consists of an input layer with 14 neurons, one hidden layer with 12 neurons, and an out layer with one neuron. Considering the fact that the solar radiation of each time slot is relatively correlated to the past few hours of that time slot as well as the same time slot for past few days, we consider inputs of the network be from the past 10 samples of solar data (i.e., past 5 hours) as well as 4 samples from 4 previous days, i.e., 14 samples overall. For example, when each energy rate sample is generated every 30 minutes, to predict the energy rate for 1 PM to 1:30 PM, the data from 8 AM to 1 PM of the same day (10 samples) as well as the data of 1 PM to 1:30 PM from the past 4 days (4 samples) is used as an input to the network. In the hidden layer $Relu$ transfer function is used for all the neurons.

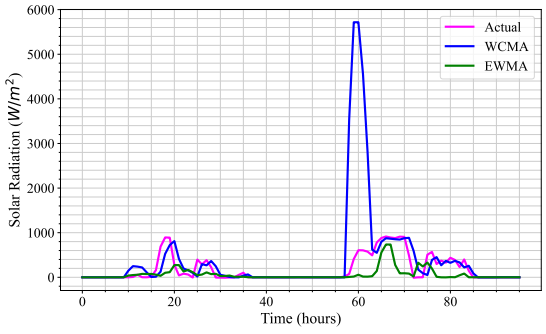

Fig. 3. Existing methods fail to predict robustly across different days

B. Task Scheduler

In most of the sensing applications, maintaining a desired data freshness of tasks has higher importance than increasing number of executed jobs or reducing the number of deadline misses during a specific time. For periodic task scheduling on batteryless devices in overload cases, i.e. when the charging rate of the system decreases, the deadline misses are unavoidable. However, finding an efficient scheduling method to adapt and maintain a required data freshness of the system with different charging rates is still a challenge. Two possible approaches we consider in this paper are skipping some jobs or adjusting the period of tasks. We explore how these two different approaches can be used in batteryless devices with conventional real-time scheduling policies such as EDF and introduce an alternative policy focusing on AoI.

Lemma 1: According to [4], for a taskset with n tasks ordered by their relative deadlines, i.e. , $\forall i, j \leq n, i \leq j \rightarrow D_i \leq D_j$, the taskset is schedulable with the EDF scheduling policy if

$$\forall k = 1, ..., n, \quad \sum_{i=1}^{k} \left(\frac{C_i + Q_i^+}{D_i} \right) + \frac{B_k}{D_k} \leq 1 \qquad (3)$$

where $Q_i^+ = max\{Q_i, 0\}$. In the above equation, B_k is the blocking time of a task τ_k that can be obtained by

$$B_k = \max_{j:D_j > D_k} C_j \qquad (4)$$

Proof: The equation is the based on the EDF schedulability with the Stack Resource Policy (SRP) provided in [25] with added energy constraints. Further details of the proof is elaborated in [4]. ■

It should be noted that for the Eq. (3) to hold, the deadline of each task should be always smaller than or equal to the period of the task, i.e. $\forall i = 1, ..., n, \quad D_i \leq T_i$.

Even though Lemma 1 which is proposed by [4] can be used to find the schedulability of the taskset under the EDF policy, during overload situations where the charging rate is not enough for the tasks to meet their deadlines, it cannot provide any solution although AoI constraints could still be satisfied. Furthermore, during the time when the charging rate is high enough that the taskset can run at a faster pace, the EDF scheduler can only provide a schedule just to meet the *deadlines* without further optimizing for AoI constraints.

Hence, we propose the Least AoI Slack First (LASF) policy to address AoI requirements. We also provide two scheduling methods, job skipping and elastic period adjustment, for the LASF and EDF policies. Either of these scheduling methods can be used in the system considering on the system's limitations and constraints. For all these methods, the scheduler uses the charging rate of the system, m_a, predicted by the energy predictor and tries to minimize the average age of information of the taskset considering the current charging rate

of the system as well as the required data freshness given by the AoI constraint of each task, MTA.

In the LASF policy, the remaining tolerable AoI of tasks, i.e., $\arg\min_i (ASD_i - A_i)$ is used to find the highest priority task, where A_i is the AoI of task τ_i and ASD_i is the AoI deadline of task τ_i which can be calculated as

$$ASD_i = \widehat{MTA}_i - C_i \tag{5}$$

We define $\widehat{MTA}_i$ as the pseudo maximum tolerable AoI of task τ_i that can be calculated as

$$\widehat{MTA}_k = U_l \times MTA_k \tag{6}$$

where U_l is the utilization bound to satisfy the data freshness given by AoI constraint MTA, i.e. if $U_l > 1$, the required data freshness cannot be satisfied. U_l can be obtained by

$$U_l = \max_{k \le n} \left\{ \sum_{i=1}^{k} \left(\frac{C_i + Q_i^+}{MTA_i} \right) + \frac{B_k}{MTA_k} \right\} \tag{7}$$

It should be noted that even in the case when data freshness cannot be satisfied, i.e., $U_l > 1$, this scheduling policy tries to minimize the AoI of the taskset.

1) Job Skipping: During overloads on the system, one of the approaches can be used is skipping some of the jobs. When the EDF policy is used with the job skipping method, we skip jobs of a task if the previous job of the same task has not finished its execution. That is, the job queue of the scheduler can only contain one job from each task at each time. It should be noted that each job can only enter the job queue at its release time, therefore, ignored jobs will not be considered to execute at any time.

The job skipping method for the $LASF$ policy is described in Algorithm 1. In this method, we first find a utilization bound for the given AoI constraints from (7). Then, we generate $\widehat{MTA}$ and ASD for each task, shown in line 4 to 7. Then the ASD and the current AoI of tasks, A_i, are used to find the highest priority task τ_h shown on line 8 of the algorithm. If the current energy ($Curr\_Charge$) is enough to execute the highest priority task, then we execute it. Otherwise, we find the next scheduling decision time, t_{new}, shown on line 12 to 15 of Algorithm and put the device into sleep mode for the difference of current time and t_{new} to preserve energy during this time. As discussed earlier, only one job of each task can enter the scheduling queue. Therefore, jobs of a task are skipped at their arrival time if the previous job of the same task is not executed.

2) Elastic Period Adjustment: There are systems that allow changing the period of tasks at run time. For these type of systems, we present an elastic period adjustment method. In this method, we change the period of tasks so that the arrival rate of jobs adapts to the predicted charging rate of the system. For the EDF policy with elastic period adjustment, we change the period of task τ_i to $T_i' = T_i \times U_e$, where U_e is the utilization bound based on the taskset deadlines and can be calculated as

$$U_e = \max_{k \le n} \left\{ \sum_{i=1}^{k} \left(\frac{C_i + Q_i^+}{T_i} \right) + \frac{B_k}{T_k} \right\} \tag{8}$$

Based on Lemma 1, this guarantees the schedulability of the taskset given the deadline of each task to be equal to its period as:

$$\begin{aligned} \max_{k \le n} \left\{ \sum_{i=1}^{k} \left(\frac{C_i + Q_i^+}{T_i'} \right) + \frac{B_k}{T_k'} \right\} = \\ \frac{1}{U_e} \max_{k \le n} \left\{ \sum_{i=1}^{k} \left(\frac{C_i + Q_i^+}{T_i} \right) + \frac{B_k}{T_k} \right\} = 1 \end{aligned} \tag{9}$$

For the LASF policy, we set periods as $T_i' = \widehat{MTA}_i$ and then use Algorithm 1 to find the schedule. This guarantees each task τ_i to meet its AoI constraint of $\widehat{MTA}_i$.

We later compare the performance of each method in term of average data freshness.

Algorithm 1 Least AoI Slack First

1: $t \leftarrow$ current time
2: Update A_i of each task
3: Compute U_l by (7)
4: **for** $k \le n$ **do**
5: $\quad \widehat{MTA}_k = U_l \times MTA_k$ ▷ U_l is obtained by (7)
6: $\quad ASD_i = \widehat{MTA}_i - C_i$
7: **end for**
8: $h \leftarrow \arg\min_i (ASD_i - A_i)$
9: **if** $Curr\_Charge \ge Q_i$ **then**
10: $\quad$ Execute the task τ_i
11: **else**
12: $\quad t_r \leftarrow$ earliest release time of a job from any of the tasks
13: $\quad t_c \leftarrow t + Q_i - Curr\_Charge$
14: $\quad t_{new} \leftarrow min\{t_c, t_r\}$
15: $\quad t \leftarrow t_{new}$ ▷ Device goes to sleep for $t_{new} - t$ seconds
16: **end if**

V. Evaluation

A. Energy Prediction

In this section we provide the evaluation results from proposed NN-based predictor as well as state of the arts, i.e. EWMA, WCMA, and Pro-Energy. We both compare the the prediction error as well as the runtime overhead of the predictors. We use solar data collected from NREL database [24] to compare the performance of each method. All the methods were implemented in python and were performed on Raspberry Pi 3 (RPI3) as an embedded system example. In this experiment, we divide the solar data into 30 minutes time slots and in case of multiple samples during the time slot, we average the energy reception rate for each time slot. We use the solar data from 2017 to 2019 to train the networks. The prediction results were conducted for the entire year of 2020. We found that the accuracy is not the proper metric to measure the performance of the predictor since there are a lot of small and zero values in the data, especially during the night and cloudy days. Hence, we use average prediction error for each of the methods to compare the performance. We conducted the prediction data for the entire year of 2020, i.e. $365 * 48 = 17520$ samples, for each of the methods, then measured the Mean Absolute Error (MAE) of the prediction compared to the ground truth, and averaged the error over the number of predicted samples. Fig. 4 shows the average error for different predictors for the entire year of 2020. For

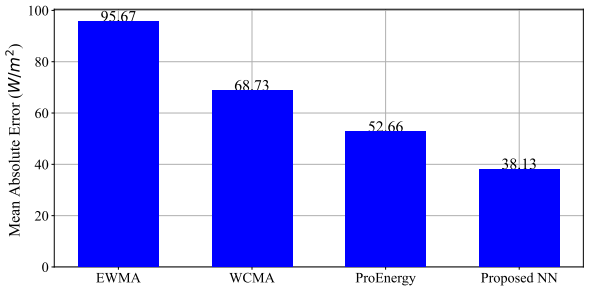

Fig. 4. Predictors correctness comparison

reference, a regular sunny day in the dataset has an average solar energy rate of about 800 to 1200 $Watt/m^2$ at noon. As shown in Fig. 4, the proposed predictor outperform all the previous methods.

One of the important aspects of an energy predictor for batteryless devices is the processing overhead it adds to the system. We conduct another experiment to evaluate the average runtime of each of the methods. Similar to the previous evaluation, we measured the execution time of each method for the entire year of 2020 and averaged the run time over the number of generated samples. Fig. 5 shows that proposed method outperformed all of the previous methods, having very low execution time overhead.

Although EWMA still has the lowest runtime due its simplicity, the accuracy gain achieved by the proposed predictor is significant considering to its slightly higher runtime overhead. It should be noted that although the runtime of all the methods are in the order of milliseconds, the embedded system used in this experiment is relatively powerful with the processor frequency of 1.4GHz, whereas most batteryless devices are equipped with microcontrollers with much lower frequencies and thus make the runtime overhead of the predictor much more significant.

B. Task scheduling

We conduct multiple scheduling experiments to compare the performance of the methods presented in Section IV-B under different scenarios. For each task, to generate task execution time, the taskset utilization is obtained using the UUniFast method [26], multiplied by the task periods, and then rounded to the nearest multiple of 0.05, i.e., $C_i = \max(\lfloor \frac{T_i}{0.05} \cdot U_i \rceil, 1) \times 0.05$. In all of our experiments, all the tasks in the taskset are

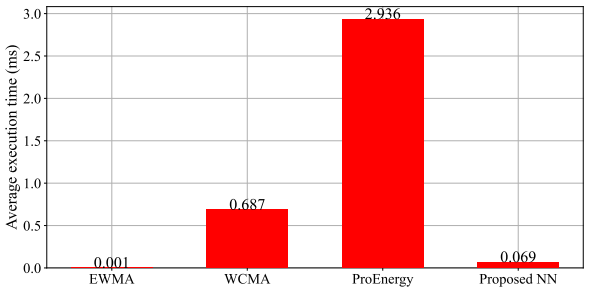

Fig. 5. Predictors runtime overhead comparison

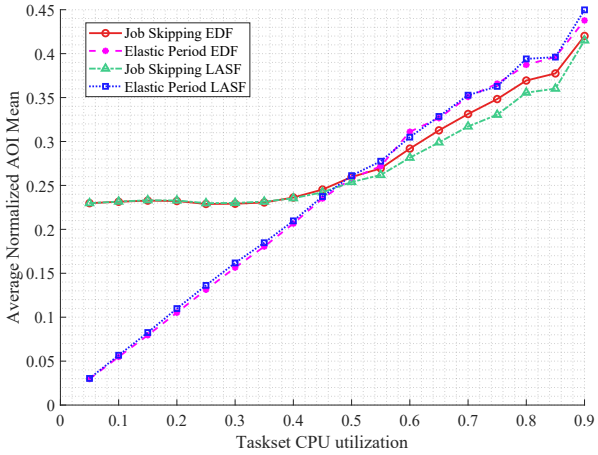

Fig. 6. Schedulers AoI performance for different taskset utilization

considered to be sensing tasks and data of the sensors are stored locally on non-volatile memory of the sensing device.

We first evaluate the effect of taskset utilization on average age of information. In this experiment, the taskset utilization is chosen from 0.05 to 0.9 in 0.05 steps. For each taskset utilization, 1000 tasksets are generated and the average of normalized AoI mean of the total 1000 tasksets is reported. Specifically, for each point, we calculate the following:

$$\hat{\mu}_A = \frac{\sum_{i=1}^{1000} \left(\frac{\sum_{\gamma=1}^{N^\gamma} (\mu_{A_i^\gamma} / MTA_i^\gamma)}{N^\gamma} \right)}{1000} \tag{10}$$

where $\mu_{A_i^\gamma}$ and MTA_i^γ are the average age of information and the maximum tolerable age of information of task τ_i in the taskset γ containing N^γ tasks, respectively. It should be noted that MTA is usually defined more than period but based on the system's data freshness requirements. $\mu_{A_i^\gamma}$ can be calculated as described in Section III-A.

The number of tasks, the period, and the discharging rate of each task are chosen randomly from 2 to 10, from 1s to 50s, and from 1 to 10, respectively (all in integers). The MTA for each task is also chosen randomly from 1x to 4x of its period. The charging time of each task, Q_i, can be calculated from (2). In all experiments, the charging rate is fixed to 3, and deadlines are set equal to task periods ($D_i = T_i$). Fig. 6 shows the performance of different scheduling methods at different taskset utilizations. As shown in the figure, the LASF policy with elastic period adjustment performs better for lower utilization tasksets, while LASF with job skipping works better for higher utilization tasksets. The figure depicts that LASF with job skipping has better performance than EDF with job skipping, but there is no significant difference between the EDF and LASF policies when elastic period adjustment is used. The similar performance of LASF and EDF with elastic period adjustment can be due to the fact that MTA is chosen as multiples of period; hence, it causes new task periods under the two policies to be often the same, resulting in similar task scheduling behavior in overload situations.

To evaluate the effect of discharging rate of tasks in the taskset on average age of information, we set up another

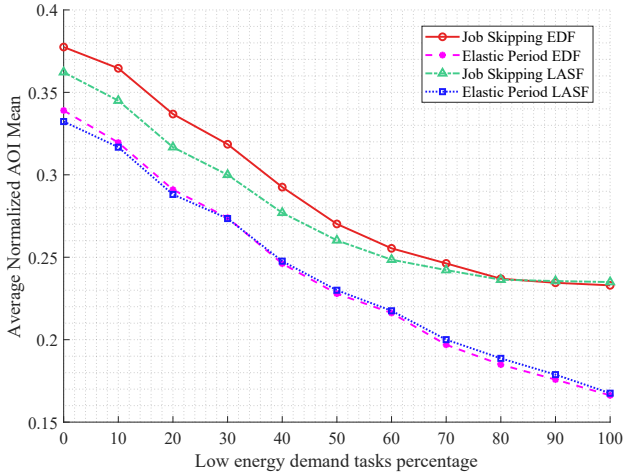

Fig. 7. Schedulers AoI performance for different taskset sizes

experiment which we divide tasks into two categories: high and low energy demands. The discharging rate is randomly chosen from 8 to 10 for high energy demand tasks, and from 1 to 3 for low energy demand tasks. The rest of the parameters are chosen similar to the previous experiment. Fig. 7 depicts the performance of each method. As shown in Fig. 7 the proposed LASF has noticeable improvement over EDF on job skipping method. However, on elastic period method, there is no significant difference between EDF and LASF policies.

VI. Conclusion

In this paper, we presented a new framework for scheduling real-time sensing tasks with data freshness constraints on batteryless devices. We proposed a lightweight machine learning based solar energy predictor which can be easily implemented on microcontrollers. Our proposed predictor outperformed the stat-of-the-art methods in terms of mean absolute error as well as runtime overhead. We also studied job skipping and elastic period adjustment scheduling methods to deal with overload situations where the charging rate decreases, and proposed the LASF policy as an alternative to the standard EDF policy. For future work, the combination of tasks with AoI constraints and hard deadlines can be considered. Such tasksets can contain preprocessing tasks before running sensing tasks or transmission tasks to send output to a remote device over low power communication such as Bluetooth Low Energy (BLE). We plan to explore these issues and further evaluate our methods in real-world conditions.

Acknowledgment

This work was sponsored by the National Science Foundation (NSF) grant 1943265, the National Institute of Justice (NIJ) grant 2019-NE-BX-0006, and the National Institute of Food and Agriculture (NIFA) grant 2020-51181-32198.

References

[1] A. P. Sample, D. J. Yeager, P. S. Powledge, A. V. Mamishev, and J. R. Smith, "Design of an RFID-based battery-free programmable sensing platform," *IEEE Trans. on Inst. and Measurement*, vol. 57, no. 11, pp. 2608–2615, 2008.

[2] A. Colin, E. Ruppel, and B. Lucia, "A Reconfigurable Energy Storage Architecture for Energy-harvesting Devices," in *ACM SIGPLAN Notices*, vol. 53, 2018.

[3] J. Hester and J. Sorber, "Flicker: Rapid prototyping for the batteryless internet-of-things," in *SenSys*, 2017.

[4] M. Karimi, H. Choi, Y. Wang, Y. Xiang, and H. Kim, "Real-Time Task Scheduling on Intermittently Powered Batteryless Devices," *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13 328–13 342, 2021.

[5] M. Karimi and H. Kim, "Energy Scheduling for Task Execution on Intermittently-Powered Devices," *ACM SIGBED Review*, vol. 17, no. 1, pp. 36–41, 2020.

[6] Y. Takatsuka, H. Nagao, T. Yaguchi, M. Hanai, and K. Shudo, "A caching mechanism based on data freshness," in *2016 International Conference on Big Data and Smart Computing (BigComp)*, 2016, pp. 329–332.

[7] T. Yamashita, "Distributed View Divergence Control of Data Freshness in Replicated Database Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 10, pp. 1403–1417, 2009.

[8] B. Islam and S. Nirjon, "Scheduling computational and energy harvesting tasks in deadline-aware intermittent systems," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2020.

[9] K. S. Yildirim, A. Y. Majid, D. Patoukas, K. Schaper, P. Pawelczak, and J. Hester, "InK: Reactive Kernel for Tiny Batteryless Sensors," in *SenSys*, 2018.

[10] K. Maeng and B. Lucia, "Adaptive low-overhead scheduling for periodic and reactive intermittent execution," in *PLDI*, 2020.

[11] D. K. Noh and K. Kang, "Balanced energy allocation scheme for a solar-powered sensor system and its effects on network-wide performance," *J. Comput. Syst. Sci.*, vol. 77, no. 5, p. 917–932, Sep. 2011.

[12] J. Recas Piorno, C. Bergonzini, D. Atienza, and T. Simunic Rosing, "Prediction and management in energy harvested wireless sensor nodes," in *International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology*, 2009, pp. 6–10.

[13] A. Cammarano *et al.*, "Pro-energy: A novel energy prediction model for solar and wind energy-harvesting wireless sensor networks," in *IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, 2012.

[14] S. Al-Alawi and H. Al-Hinai, "An ann-based approach for predicting global radiation in locations with no direct measurement instrumentation," *Renewable Energy*, vol. 14, no. 1, pp. 199–204, 1998.

[15] Y. Jiang, "Prediction of monthly mean daily diffuse solar radiation using artificial neural networks and comparison with other empirical models," *Energy Policy*, vol. 36, no. 10, pp. 3833–3837, 2008.

[16] A. Koca, H. F. Oztop, Y. Varol, and G. O. Koca, "Estimation of solar radiation using artificial neural networks with different input parameters for mediterranean region of anatolia in turkey," *Expert Systems with Applications*, vol. 38, no. 7, pp. 8756–8762, 2011.

[17] A. Sharafati *et al.*, "The potential of novel data mining models for global solar radiation prediction," *International Journal of Environmental Science and Technology*, vol. 16, no. 11, p. 7147–7164, 2019.

[18] Z. Pang, F. Niu, and Z. O'Neill, "Solar radiation prediction using recurrent neural network and artificial neural network: A case study with comparisons," *Renewable Energy*, vol. 156, pp. 279–289, 2020.

[19] Y. Ge, Y. Nan, and L. Bai, "A hybrid prediction model for solar radiation based on long short-term memory, empirical mode decomposition, and solar profiles for energy harvesting wireless sensor networks," *Energies*, vol. 12, no. 24, 2019.

[20] S. Kosunalp, "A new energy prediction algorithm for energy-harvesting wireless sensor networks with q-learning," *IEEE Access*, vol. 4, pp. 5755–5763, 2016.

[21] M. Chu, H. Li, X. Liao, and S. Cui, "Reinforcement learning based multi-access control and battery prediction with energy harvesting in iot systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2009–2020, 2018.

[22] J. Hester, K. Storer, and J. Sorber, "Timely Execution on Intermittently Powered Batteryless Sensors," in *SenSys*, 2018.

[23] K. Maeng and B. Lucia, "Adaptive dynamic checkpointing for safe efficient intermittent computing," in *OSDI*, 2018.

[24] A. M. Andreas, "University of nevada - las vegas," 2006. [Online]. Available: https://midcdmz.nrel.gov/apps/sitehome.pl?site=UNLV

[25] T. P. Baker, "Stack-based scheduling of realtime processes," *Real-Time Systems*, vol. 3, no. 1, pp. 67–99, 1991.

[26] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129–154, 2005.