# Battery Aging Deceleration for Power-Consuming Real-Time Systems

Jaeheon Kwak*, Kilho Lee*, Taehee Kim*, Jinkyu Lee†, Insik Shin*
*School of Computing, KAIST, Republic of Korea
†Department of Computer Science and Engineering, Sungkyunkwan University (SKKU), Republic of Korea
jinkyu.lee@skku.edu

***Abstract*—Battery aging is one of the critical issues in battery-powered electric systems. However, this issue has not received much attention in the real-time systems community. In this paper, we present the first attempt to translate the problem of minimizing battery aging subject to timing requirements into a real-time scheduling problem, addressing the following issues. (i) Can scheduling make a systematic impact on battery aging? If so, which scheduling principles are favorable to minimizing battery aging? (ii) If there exists any, how can we build upon the scheduling principle to guarantee real-time requirements? For (i), we first illuminate the connection between task scheduling and battery aging minimization and then derive a principle for task scheduling from abstracting the complicated dynamics of battery aging, which is to minimize the variance of total power consumption over time. In addition, we implement a battery aging simulator and use it to verify the effectiveness of the proposed principle in minimizing battery aging and its impact on quantitative improvement. For (ii), we propose a scheduling framework that separates control for timing guarantees from that for battery aging minimization. Such a separation allows reducing the complexity significantly such that we can employ existing scheduling algorithm and schedulability analysis for real-time guarantee and tailor the proposed scheduling principle to decelerate battery aging without taking real-time guarantees into accounts. Our simulation results show that the proposed framework can extend the battery lifespan by up to 144.4%.**

## I. Introduction

These days, each subsystem in an electric system often accommodates a set of real-time power-consuming tasks, which typically shares the system-wide power source while having different characteristics of power consumption and timing requirements. To support mobility, a battery pack (typically Li-ion batteries [1]) becomes a popular power source for a wide range of electric systems, such as EVs (Electric Vehicles), satellites, spacecraft and UAVs (Unmanned Aerial Vehicles) as shown in Figure 1 [2]–[4]. Since it is impossible or costly to replace worn-out batteries in those electric systems, it is crucial to minimize battery aging [3], [5] to prolong the lifetime of not only the battery pack but also sometimes the entire system itself. As an example, Euclid [5], a space telescope to be launched in 2022, is equipped with a power system consisting of solar panels and Li-ion battery cells. Since this system not only disallows to replace the battery cells during its mission but also has stringent space and weight constraints, it is very important to minimize the aging of battery cells.

†Jinkyu Lee is the corresponding author.

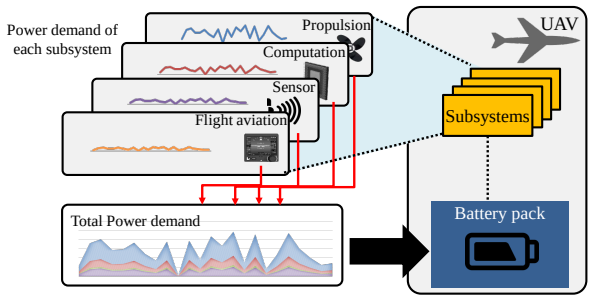



Fig. 1: Overview of a battery-powered electric system consisting of power-consuming real-time subsystems

The electric systems explained so far have the following two requirements:

R1. Offering timing guarantees of all real-time power-consuming tasks belonging to each subsystem, and

R2. Minimizing battery aging of a pack of batteries that supplies power shared by the subsystems.

While there have been numerous studies that successfully address *either* R1 *or* R2, those studies cannot achieve both requirements at the same time. For example, plenty of studies in the real-time systems community focused on R1 without regarding R2, lacking the development of effective scheduling strategies from a comprehensive understanding of battery characteristics and aging. On the other hand, studies for R2 in the battery systems community have paid little attention to timing guarantees when minimizing battery aging. The most relevant studies from the R1 side are a number of studies that minimize power consumption or peak power without compromising timing guarantees [6]–[8]. Although successful in achieving power-related goals, they cannot be applied to the electric system of interest in this paper because achieving power-related goals is different from minimizing battery aging. When it comes to R2, there are only a few studies addressing timing guarantees with battery-related goals [9]–[11]; however, they have not focused on battery aging.

In this paper, we aim at achieving R1 and R2 at the same time for battery-powered electric systems. To this end, we take an unprecedented point of view, which leverages real-time task scheduling to address R1 and R2 simultaneously, which entails the following questions.

C1. Is it possible for task scheduling to affect battery aging? If so, what is a scheduling principle favorable to minimizing battery aging?

C2. Once C1 is addressed, how can we develop a scheduling methodology that not only addresses R1 and R2 together but also exhibits low complexity for offline/run-time operations?

To achieve R1 and R2 together by addressing C1 and C2, our approach is outlined as follows.

A1. Based on a deep understanding of battery behavior, we derive a core scheduling principle to minimize battery aging.

A2. To follow the principle with an affordable overhead, we propose a scheduling framework that separates control for timing guarantees from that for battery aging minimization.

A1 is detailed as follows. We abstract the complicated dynamics of battery aging and analyze the relationship between task scheduling and battery aging. We then derive a basic scheduling principle: minimizing the variance of current load changes. To support this, we implement a battery aging simulator, which can be utilized to validate the feasibility of task scheduling towards minimizing battery aging and the effectiveness of the proposed principle.

When it comes to A2, we propose the `RET` scheduling framework that offers offline timing guarantees by the existing schedulability analysis while achieving battery aging minimization through runtime scheduling heuristics favorable to battery aging. To this end, `RET` operates as follows. In the offline stage, `RET` assigns an inflated execution time (called the *reserved execution time*) without compromising offline timing guarantees. At runtime, `RET` reserves the subsystem (instead of performing its execution on the subsystem) when scheduled by the target scheduling algorithm; it then determines when the actual execution is performed within the reserved execution interval, to minimize battery aging. On top of this architecture, we propose two effective heuristics exploiting scheduling control knobs offered by each part. This divide-and-conquer approach not only reduces the problem complexity but also enables to utilize well-studied existing real-time scheduling theories.

We demonstrate the effectiveness of the framework through the proposed battery aging simulator. Our simulation results show that the scheduling principle from A1 exhibits a strong relationship with battery aging. In addition, `RET` effectively increases the battery lifespan regardless of the system setup, i.e., up to 144.4% of improvement compared to the baseline EDF (Earliest Deadline First) scheduler.

In summary, this paper makes the following contributions:

- We introduce a scheduling principle for decelerating battery aging based on a deep understanding of electrochemical characteristics of batteries and verify the effectiveness of the principle through our own simulator (Section III).
- We develop a novel scheduling framework which achieves timing guarantee and battery aging minimization at the same time (Sections IV and V). To the best of our knowledge, this is the first work to consider both issues at the same time.
- We propose effective ways of exploiting control knobs offered by the framework in accordance with the scheduling principle that decelerates battery aging (Section VI).
- We present extensive evaluations with an accurate battery simulator implemented based on the widely used real-world battery (Section VII).

In addition to the sections mentioned with the contributions, the rest of this paper is structured as follows. Section II presents our system model, and Section VIII summarizes related work. Finally, Section IX discusses two interesting issues, and then Section X concludes the paper.

## II. System Model

In this paper, we target an electric system $\mathcal{S}$, consisting of $n$ subsystems $\{\mathcal{S}^j\}_{j=1}^n$. Each subsystem $\mathcal{S}^j$ accommodates a set of real-time power-consuming tasks $\tau^j = \{\tau_i^j\}$. We assume that each real-time power-consuming task is sporadic/periodic and non-preemptive, and has an implicit deadline. That is, each real-time power-consuming task $\tau_i^j \in \tau^j$ has three parameters $(T_i^j, C_i^j, P_i^j)$, where $T_i^j$ is the minimum separation (or period) between successive execution requests, $C_i^j$ is the worst-case execution time (WCET), and $P_i^j$ is the power consumption during execution.[1] Note that each $\tau_i^j$ generates potentially unbounded series of execution requests. Once $\tau_i^j$ requests its execution, the subsystem $\mathcal{S}^j$ should finish $\tau_i^j$'s execution within $T_i^j$ time units from its request. Whenever the subsystem $\mathcal{S}^j$ starts to execute $\tau_i^j$, the execution cannot be preempted during $C_i^j$ time units and consumes $P_i^j$ amount of power during the execution. Each subsystem allows only one task to perform its execution at a time. Then, except for power consumption, the task model associated with each subsystem is equivalent to the implicit-deadline sporadic/periodic non-preemptive real-time task model [12], [13] on a uniprocessor platform. In this paper, we assume a quantum-based time and let one time unit be a quantum length without loss of generality; therefore, every task parameter is an integer value.

The system model described so far sufficiently reflects real-world electric systems. For example, UAVs and EVs not only consist of multiple subsystems (e.g., uniprocessor, sensor, and actuator), but also perform power-consuming tasks (e.g., computing, electrical, and mechanical tasks) corresponding to each subsystem. When it comes to an example of their subsystems, a motor subsystem (denoted by $\mathcal{S}^j$) can run forward/backward rotations as its tasks (denoted by $\tau_i^j$); the tasks are power-consuming tasks that compete for the chance to perform within the motor subsystem, disallow to get preempted by other

[1] For simplicity, we assume no or negligible power consumed during the idle state. If this does not hold, we simply add the corresponding base power to every time slot, which does not compromise the validity of the proposed solution to be presented throughout the paper.

tasks, and execute in a sporadic/periodic manner. Therefore, it is reasonable to model the tasks as sporadic/periodic non-preemptive tasks on a subsystem that allows only one task to perform its execution at a time.

With regard to a power supply, all the subsystems share the electric power of the target electric system, which is supplied by a single battery pack consisting of multiple batteries and controlled by BMS (Battery Management System) to equally charge/discharge all batteries. Since the battery pack provides a great abstraction to use the battery cells therein, in this paper, we consider the battery pack as a single battery. In particular, for battery-specific parameters, we consider a pack of multiple 26650 sized 2.3Ah LFP (Lithium Iron Phosphate) battery cells manufactured by A123 system [14], [15], which belong to one of the most widely used battery types and are used in many existing studies. Despite the target battery cell type, the battery model and the scheduling framework to be proposed in the rest of the paper can be generally applicable to most (if not all) types of battery cells.

## III. Battery Aging: Understanding and Connecting with Task Scheduling

In this section, we present details of one of the two main contributions of this paper, which establishes the foundation of task scheduling for battery aging minimization. To this end, we first investigate an existing battery aging model and derive a task scheduling strategy that yields battery aging minimization, from the investigation. We then present our implementation of the battery aging simulator, which can be utilized for solving potential task scheduling problems relevant to battery aging. Finally, we present motivation experiments via the simulator, which validate not only the feasibility of task scheduling towards minimizing battery aging, but also the effectiveness of the proposed task scheduling strategy.

### A. Understanding of Battery Aging towards Task Scheduling

A battery is an electrochemical device containing chemical energy which can be readily converted to electrical energy. Capacity of a battery refers to the total amount of electronic charge the battery delivers when the battery is fully discharged in one hour. A Li-ion battery typically loses its capacity during charge, discharge, and even rest. The effect of losing capacity is called capacity degradation (also called battery aging); once a battery loses its capacity, the capacity never can be recovered. Generally, if a battery loses 20% of its capacity, it is regarded as a dead battery and recommended to be replaced.

Many studies have been proposed for battery aging modelings, such as empirical model [16], circuit model [17] and electrochemical model [14]. Among them, we focus on the electrochemical model [14], which is one of the most accurate models and arguably the only model that can explain battery aging phenomenon intuitively [18].

According to this model, battery aging is affected by two major reasons: active material loss and SEI (Solid Electrolyte Interface) layer growth. Thereby, battery aging ($Q$) can be represented in terms of the percentage of lost capacity, and calculated as the sum of the lost capacity incurred by the active material loss ($Q_{\text{AM}}$) and the SEI layer growth ($Q_{\text{SEI}}$) as follows:

$$Q = Q_{\text{AM}} + Q_{\text{SEI}},$$

$$Q_{\text{AM}} = \int_0^L k_{\text{AM}} \cdot \exp(-\frac{E_{\text{AM}}}{R_{gas} \cdot T}) \cdot SOC \cdot |I| \; dt, \quad (1)$$

$$Q_{SEI} = \int_0^L -\frac{k_{SEI}}{2\sqrt{t}} \cdot \exp(-\frac{E_{\text{SEI}}}{R_{gas} \cdot T}) \; dt, \quad (2)$$

where $k_{\text{SEI}}$, $k_{\text{AM}}$, $E_{\text{SEI}}$, $E_{\text{AM}}$ and $R_{gas}$ denote some constants, respectively, and $L$ denotes any length of a time interval. Note that we assume that side reaction is limited by diffusion (according to the diffusion limited model [19]). Within the interval of length $L$, we may control three factors: charge/discharge current load ($I$), state of charge ($SOC$), and temperature ($T$). We now investigate how each factor controlled by task scheduling affects battery aging.

We first figure out that it is difficult to minimize battery aging by controlling $I$ and $SOC$ in Eq. (1). The charge/discharge current load $I$ of the battery directly depends on task scheduling. In our system model, each task $\tau_i^j$ consumes power $P_i^j$; thereby, based on the task scheduling, the system-wide power consumption $P$ of each time instant widely varies. In addition, since we assume that the voltage of the battery is constant over time, the current load $I$ is proportional to the power $P$ (i.e., $P = I \cdot V$, where $V$ denotes voltage). Therefore, the scheduler can control the current load $I$ of each time instant. However, the scheduler cannot control the cumulative current load (i.e., $\int I \; dt$) for a long interval (e.g., a hyper-period of all tasks), since a given set of tasks consume power collectively to complete their execution. Due to this characteristic, the scheduler cannot decelerate battery aging by controlling the current load factor $|I|$. Similarly, it is difficult to control the state of charge factor $SOC$ (in Eq. (1)) for the scheduler, since $SOC$ remains almost constant for a relatively short time interval.

We then focus on the temperature factor $T$ which affects both aging causes, $Q_{\text{AM}}$ and $Q_{\text{SEI}}$. And, we confirm that $T$ is critical to battery aging. If we apply Taylor expansion based on room temperature (25°C) to Eqs. (1) and (2), lowering 1°C temperature results about 5.3% and 5.4% decrease in $Q_{\text{AM}}$ and $Q_{\text{SEI}}$, respectively. In addition, task scheduling can easily control the temperature, since the temperature depends on the current load $I$. The temperature $T$ is determined by the heat of the battery which is calculated as follows [20]:

$$\rho c_p \frac{dT}{dt} = q_{gen} + q_{conv}, \quad (3)$$

where $\rho c_p$, $q_{gen}$ and $q_{conv}$ denote a constant, the heat generated from the battery itself, and the heat transferred by

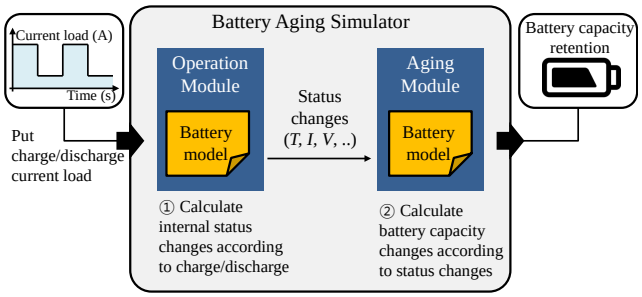



Fig. 2: Battery simulator architecture

convection, respectively. We focus on $q_{gen}$, the major factor of an increase in the temperature. In a battery, $q_{gen}$ consists of a number of heat sources including reversible heat $q_{\text{rev}}$, irreversible heat $q_{\text{irr}}$, and ohmic heat $q_{\text{ohm}}$. Among those three sources, the most dominant factor is known as ohmic heat [21], [22], which is represented as:

$$q_{\text{ohm}} = I^2 \cdot R_c, \tag{4}$$

where $R_c$ denotes one of the internal resistances of a battery (current collector resistance) which has a constant value. According to Eqs. (3) and (4), for a given interval of any length $L$, the scheduler can minimize the temperature $T$ of the battery in the interval, by minimizing the square sum of current load $I$ over the interval. In summary, we confirm that the temperature $T$ is the most critical factor for minimizing battery aging since it significantly affects both $Q_{\text{AM}}$ and $Q_{\text{SEI}}$ and it can be easily managed by task scheduling.

Based on the understanding of the battery dynamics, we now derive a basic scheduling principle favorable to battery aging: minimizing the variance of current load changes. Since the heat and the temperature mostly depend on the sum of squared current $I^2$, we can minimize battery aging by minimizing $I^2$. For a more intuitive principle, we consider minimizing the variance of $I$, which is calculated as $E(I^2) - E(I)^2$, where $E(x)$ denotes the expected value of $x$. As aforementioned, $E(I)$ is fixed within a given interval $L$; thus, we can translate minimizing $I^2$ into minimizing the variance of $I$. Also, note that minimizing the variance of $I$ is equivalent to minimizing the variance of the power consumption $P$.

### B. Battery Aging Simulator

To closely investigate the effect of different battery usage on battery aging, we implemented an accurate battery aging simulator as shown in Figure 2. The simulator takes battery current load changes over time as input. After that, *Operation Module* calculates the battery internal status changes according to the current load; then, *Aging Module* calculates the battery capacity changes according to the status changes and returns the remaining capacity as a battery aging result.

To maximize the accuracy of the simulation, we implemented both modules based on an electrochemical model known as the most accurate one [22]. For *Operation Module*, we adopted the thermal-electrochemical P2D battery model proposed by Bizeray *et al*. [20], which is known to precisely calculate battery internal status including voltage and heat generation. Note that we implemented this module based on the open-source project [23]. For *Aging Module*, we implemented the capacity degradation model for graphite anode LIBs proposed by Jin *et al*. [14], which is known to accurately calculate capacity changes. We also tuned the models with accurate parameters for our target battery system (i.e., 26650 sized 2.3Ah LFP type), as reported by Zhang *et al*. [15].

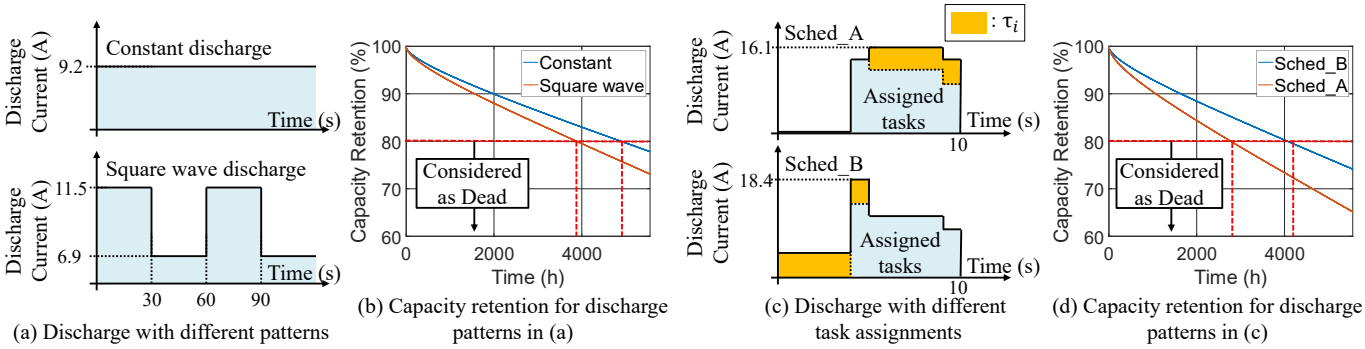
### C. Motivation Experiments

In this section, we leverage our simulator to conduct motivational experiments to answer the following questions. (Q1) Is it actually possible for different task schedules to yield different battery aging deceleration? (Q2) Is the task scheduling strategy presented in Section III-A effective in minimizing battery aging? (Q3) If so, how much can we decelerate battery aging with the task scheduling strategy?

**Simulation setup.** In each simulation, a battery repeatedly discharges and charges during sufficiently long time (i.e., around 5500 hours) to observe its capacity retention lower than 80% (which is regarded as a dead battery). In each discharge/charge cycle, the battery discharged with a given discharge pattern until the charged capacity runs out, and then the battery charges with a constant current of $2.3A$. We estimated battery capacity retention over time under different task schedules; since the capacity retention of a battery decreases as the battery ages, the capacity retention is an effective measure that represents battery aging.

**Constant vs. fluctuating patterns.** We first compare two extreme discharge patterns: constant and fluctuating patterns, both of which consume the same amount of total power consumption, as shown in Figure 3(a); note that we use a square wave as a fluctuating discharge pattern, as opposed to a constant discharge pattern. Despite the identical total power consumption, we observe that different discharge patterns (due to different task schedules) result in different behaviors of capacity retention. As shown in Figure 3(b), the battery capacity retention of the constant discharge decreases more slowly than that of the square wave; as a result, the battery lifespan of the constant discharge is 26.1% longer than the square wave. This result presents the following implications for Q1, Q2, and Q3: different schedules yield different battery aging levels (even if the total power consumption is the same); the task scheduling strategy presented in Section III-A (which is the constant discharge in this simulation) is effective in minimizing battery aging; different schedules can make significant difference in decelerating battery aging.

**Different patterns with scheduling constraints.** Although it is simple to apply the task scheduling strategy presented in Section III-A (by enforcing the constant discharge), such simplicity comes from the assumption that we can employ *any* discharge pattern for given total power consumption. However, in practice, it is impossible to keep constant discharge due to scheduling constraints such as timing requirements and




(a) Discharge with different patterns (b) Capacity retention for discharge patterns in (a) (c) Discharge with different task assignments (d) Capacity retention for discharge patterns in (c)

Fig. 3: Motivation simulations: battery capacity retention (i.e., battery aging) changes over time, depending on the scheduling policies.

unpredictability of task arrivals. We now explain another simulation with scheduling constraints. As shown in Figure 3(c), *Sched_A* and *Sched_B* respectively schedule Task $\tau_i$ (colored as yellow), subject to the already assigned other tasks (colored as sky blue); note that both schedules result in the same total power consumption. Although *Sched_A* seems closer to constant than *Sched_B*, Figure 3(d) shows that *Sched_A* is less favorable than *Sched_B* in terms of battery aging. This is because, *Sched_A* and *Sched_B* yield the variance of 57.14 and 19.04, respectively; according to the task scheduling strategy presented in Section III-A, *Sched_B* is more effective in minimizing battery aging than *Sched_A* by reducing the variance of the discharge current, yielding 46.52% longer battery lifespan. Therefore, the second simulation also answers Q1, Q2, and Q3. That is, *Sched_A* and *Sched_B* yield different battery aging (addressing Q1); *Sched_B* whose variance of power consumption is smaller than *Sched_A* is more effective in minimizing battery aging (addressing Q2), yielding 46.52% longer battery lifespan (addressing Q3).

Note that the second simulation also addresses potential misunderstanding of the relationship between battery aging minimization and peak power minimization. That is, one may think that minimizing peak power (well-known for efficient battery use [7], [9]) would be effective for the battery aging; however, *Sched_B* yields less battery aging despite the higher peak power consumption than *Sched_A*.[2]

## IV. Framework Overview

In Section III, we established the basis for connecting battery aging and task scheduling. In this section, we leverage the basis to formulate the problem and discuss its challenges. We then present our approach overview to address the problem with the challenges.

**Problem statement.** The main problem addressed in this paper is to develop a scheduling methodology that addresses battery aging minimization and timing guarantees together, which is formally expressed as follows.

[2]Here, the peak power implies the peak discharge current in that the discharge power is proportional to the discharge current (considering that we used constant input/output voltage).

Given an electric system where (i) power is supplied by a battery pack and shared by its subsystems, and (ii) each subsystem accommodates a set of real-time power-consuming tasks, determine, at each time tick, which subsystems $\mathcal{S}^j \in \mathcal{S}$ execute their own tasks $\tau_i^j \in \tau^j$, subject to (a) minimizing the battery pack's aging, (b) disallowing every subsystem to execute more than one task at the same time, and (c) guaranteeing that every request for the execution of $\tau_i^j \in \tau^j$ at $t$ is completed no later than $t + T_i^j$.

It is worth to note that the scheduling strategy described in the previous section, which is favorable to battery aging minimization by minimizing the variance of the total power consumption, only matters the above constraint (a), but not the timing constraints (b) and (c). Yet, it is quite challenging to extend the scheduling strategy while satisfying all of the three constraints together and developing a new scheduling approach that addresses the above problem. To tame the complexity of the problem, we develop a new scheduling framework according to the divide-and-conquer principle; it separates the two requirements of timing guarantees *and* battery aging minimization, in a way that it addresses the timing guarantees in the offline stage and the battery aging minimization in the runtime stage. Specifically, in the offline stage, the proposed framework first increases each task's execution time as much as possible until any further increase compromises timing guarantees according to an existing schedulability test of a given target scheduling algorithm. We call the inflated execution time the *reserved* execution time. While each vanilla subsystem schedules the execution of tasks based on their execution times according to the target scheduling algorithm, each subsystem under our proposed framework first reserves the slots for the execution of tasks based on their reserved execution times according to the same scheduling algorithm. This way, no single task will experience a deadline miss as long as its actual execution is made within the reserved slots. Here, the *actual execution* means the "actual" execution of the task within its own reserved slots. Section V will detail the proposed framework.

The proposed framework offers two control knobs that can be utilized for battery aging minimization. The first control

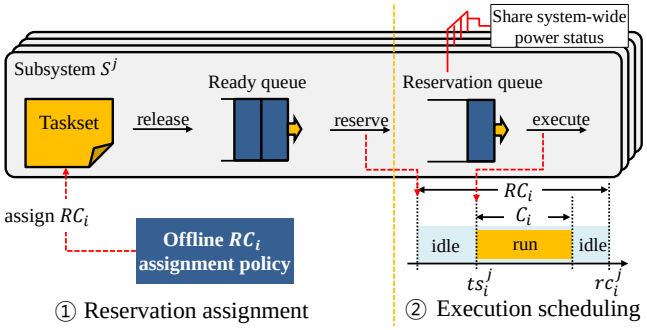



Fig. 4: *RET* framework overview

knob is to determine how much to increase the execution time of each task; while there are many Pareto optimal combinations for every task's reserved execution time, each combination yields different results for battery aging. The second control knob is to determine when to actually execute each task within its own reserved slots (whose length is determined by the first control knob). Therefore, the proposed framework can provide a basis for the development of scheduling policies on how to utilize the two control knobs towards minimizing the variance of the total power consumption. Section VI will detail how to utilize the two control knobs for battery aging minimization.

## V. The *RET* Scheduling Framework

In this section, we propose the *RET* (Reserved Execution Time) scheduling framework that provides two control knobs for battery aging minimization, while offering offline timing guarantees. As shown in Figure 4, *RET* consists of two parts: the reservation assignment and the execution scheduling. In the first (reservation assignment) part, the framework assigns the *reserved execution time* $RC_i^j$ (which is inflated from the execution time $C_i^j$) for every $\tau_i^j$ at the offline stage. At runtime, each task reserves the subsystem (instead of performing its execution) during the reserved execution time according to the given scheduling algorithm. Note that it utilizes a set of ready and reservation queues to schedule requests and reserve subsystems, respectively. In the second (execution scheduling) part, the framework schedules the actual execution (i.e., adjusts the *execution start time* $ts_i^j$) of every execution request which reserves a subsystem. The execution scheduling is determined based on the system-wide runtime power status (i.e., current load) for battery aging minimization.

Algorithm 1 details how *RET* operates. It runs with four input parameters: scheduling algorithms for each subsystem (SA), schedulability tests corresponding to SA (ST), a policy for reservation assignment (PR), and policy for execution scheduling (PE). Using the input parameters, Algorithm 1 sequentially presents the reservation assignment part in the offline stage, that in the runtime stage, and the execution scheduling part.

**Algorithm 1** *RET* (SA, ST, PR, PE)

SA: the target scheduling algorithm for each subsystem
ST: the target schedulability test for SA
PR: the policy for the reservation assignment part
PE: the policy for the execution reservation part

*Reservation assignment part—offline*:
Determine the reserved execution time $RC_i^j$ $(\geq C_i^j)$ of every task $\tau_i^j \in \tau^j$ in every subsystem $\mathcal{S}^j \in \mathcal{S}$, according the policy of PR, subject to guaranteeing by ST that every reserved execution request for each subsystem is finished before its deadline.

*Reservation assignment part—runtime*:
Whenever a new request releases or finishes,

**if** a new request of $\tau_i^j$ releases **then**
  $RQ^j \leftarrow RQ^j \cup \{\tau_i^j\}$
**end if**
**if** $|RQ^j| > 0$ **and** $VQ^j = \emptyset$ **then**
  Find $\tau_i^j$, the highest-priority task in $RQ^j$
  $VQ^j \leftarrow VQ^j \cup \{\tau_i^j\}$
  $RQ^j \leftarrow RQ^j \setminus \{\tau_i^j\}$
  $rc_i^j \leftarrow t + RC_i^j$
**end if**

*Execution scheduling part*:
Whenever a new reservation arrives or a reserved task finishes its execution,

**if** a new reservation arrives **then**
  **for** every idle $\mathcal{S}^j \in \mathcal{S}$ **do**
    Let $\tau_i^j$ denote the reserved task in $VQ^j$
    **if** $\tau_i^j$ exists **and** $t < ts_i^j$ **then**
      Update $ts_i^j$ by the policy PE
    **end if**
  **end for**
**end if**
**if** $|VQ^j| = 1$ **and** $\mathcal{S}^j$ is idle **then**
  Let $\tau_i^j$ denote the reserved task in $VQ^j$
  **if** $t = ts_i^j$ **then**
    Start actual execution of $\tau_i^j$
    Set $S^j$ as busy
  **end if**
  **if** $t = rc_i^j$ **then**
    $VQ^j \leftarrow VQ^j \setminus \{\tau_i^j\}$
  **end if**
**end if**
**if** a task of $\mathcal{S}^j \in \mathcal{S}$ completes its actual execution[3] **then**
  Set $S^j$ as idle until $rc_i^j$
**end if**

**Reservation assignment part.** This part consists of two stages: offline and runtime. In the offline stage, the framework first assigns $RC_i^j$ $(\geq C_i^j)$ of every task $\tau_i^j \in \tau^j$ in every subsystem $\mathcal{S}^j \in \mathcal{S}$, according to PR, subject to guaranteeing by ST that every reserved execution request for each subsystem is finished before its deadline. Once $RC_i^j$ of every task $\tau_i^j \in \tau^j$ in every subsystem $\mathcal{S}^j \in \mathcal{S}$ is determined offline, each subsystem schedules a set of given tasks according to SA, as

[3]For *RET*, we assume that every execution request for each task $\tau_i^j$ lasts for exactly its WCET $C_i^j$. However, *RET* also works with execution time less than its WCET; Section IX will discuss how *RET* works with execution time less than its WCET.

if each task has its execution time as $RC_i^j$. In the runtime stage, if a request of the task $\tau_i^j$ releases, the framework puts the task into $RQ^j$, where $RQ^j$ denotes the ready queue of the subsystem $\mathcal{S}^j$ (Lines 1-3). The framework then checks $RQ^j$ and $VQ^j$, where $VQ^j$ represents the reservation queue of $\mathcal{S}^j$. If there is any request in $RQ^j$, and $VQ^j$ is empty, the framework reserves $\mathcal{S}^j$ by putting a task into $VQ^j$. It moves the highest priority task (denoted by $\tau_i^j$) in $RQ^j$ to $VQ^j$ (Lines 5-7) and sets $rc_i^j$ as $RC_i^j$ time units after from the current time $t$ (Line 8), where $rc_i^j$ denotes the absolute reserved execution completion time for the request of the task $\tau_i^j$, to be used in the second part. This allows reserving the subsystem $\mathcal{S}^j$ in [$t$, $t + RC_i^j$) for that request of $\tau_i^j$.

**Execution scheduling part.** Whenever a reservation happens at any subsystem (by putting a task into the reservation queue), the framework then updates the execution start time $ts_i^j$ of every request of $\tau_i^j$ which is reserved but not executed. For this, it iterates every idle $\mathcal{S}^j \in \mathcal{S}$; and, it checks whether a request of $\tau_i^j \in VQ^j$ exists and $t < ts_i^j$ holds (Lines 2-4). If such a request exists, it updates the execution start time of that request according to the PE policy (Line 5). Note that the PE policy may communicate with other subsystems to get system-wide power status, such as a discharge power level. After that, the framework executes and finishes the reserved request. If there is a reserved request in $VQ^j$ (i.e., $|VQ^j| = 1$) and $\mathcal{S}^j$ is idle[4] (Line 9), it checks the execution start time $ts_i^j$ and the absolute reserved execution completion time $rc_i^j$. When the time tick reaches $ts_i^j$, the framework starts to execute the reserved request of $\tau_i^j$ and sets $\mathcal{S}^j$ as busy (Lines 11-14). And, when the time tick reaches $rc_i^j$, the framework finishes the reservation of $\tau_i^j$, by excluding $\tau_i^j$ from $VQ^j$ (Lines 15-17). And, the subsystem $\mathcal{S}^j$ becomes idle again if any request of $\tau_i^j$ completes its actual execution (Lines 19-21). It is worthwhile noting that each subsystem stays idle unless it performs the actual execution of an execution request. Recall we assume that each subsystem consumes no or negligible power during the idle state; thereby, the execution scheduling part can control the system-wide power consumption so as to achieve battery aging minimization.

This architecture allows separating the two main requirements (i.e., timing guarantee and battery aging minimization), and achieve them at the same time. The reservation assignment part provides freedom of the actual execution for the battery aging minimization, without compromising any timing requirements guaranteed by the offline analysis ST. This is because it guarantees to complete the $RC_i^j$ amount of reserved execution before the deadline of execution request of $\tau_i^j$ and the framework always completes the $C_i^j$ amount of actual execution within the reserved execution period (note that we enforce that PE starts the actual execution no later than $rc_i^j - C_i^j$; otherwise, the execution misses its deadline). With the timing guarantee, the execution scheduling part determines proper execution timing within the reserved execution period and provides a room for applying schedules favorable to battery aging minimization, while considering the system-wide runtime status (i.e., power load of the battery pack).

Although the *RET* framework introduces additional scheduling steps, it entails only small runtime overhead. The reservation assignment part operates the same as the target scheduling algorithm SA except reserving during $RC_i$ time units instead of executing during $C_i$ time units; thereby, the overhead of this part is comparable to that of the target algorithm SA. In the execution scheduling part, it incurs some overhead to update the execution start time $ts_i^j$ of the reserved request of $\tau_i^j$ in every subsystem $\mathcal{S}^j \in \mathcal{S}$; however, it only imposes up to $n$ (the number of subsystems) times of the PE policy computation. Since our PE policy has a small complexity as much as other global scheduling algorithms to be presented in the next section, this overhead is affordable for the scheduler.

[4] $\mathcal{S}^j$ is idle if the reserved request of $\tau_i^j$ does not start or finishes its reserved execution as shown in Figure 4.

## VI. Achieving Battery Aging Deceleration

In order to fully exploit the control knobs provided by the *RET* framework and to effectively decelerate battery aging, this section proposes two scheduling policies used as PR and PE of the framework.

### A. Reservation Assignment Part

We first present an algorithm which determines the reserved execution time of each task (i.e., $RC_i^j$ of $\tau_i^j \in \tau^j$). As shown in Algorithm 1, this algorithm is used as PR in the reservation assignment part. This part entails two different issues: 1) how to guarantee that every $\tau_i^j$ completes its reserved execution before its deadline after assigning $RC_i^j$, and 2) what is a good assignment of each $RC_i^j$ to be favorable for minimizing battery aging.

As to the first issue, we incorporate schedulability analysis into the $RC$ assignment algorithm. Since the *RET* framework schedules (reserves) $RC_i^j$ of each task according to the base scheduling algorithm SA, we can use the schedulability test ST corresponding to SA, to guarantee that every reservation with $RC_i^j$ completes before its deadline. Following our system model (see Section II), we use an uniprocessor *NP-EDF* (Non Preemptive EDF) scheduling and its corresponding schedulability analysis [12], [13]. Based on this, the analysis is as follows:

*Lemma 1 (From [12], [13]):* An implicit-deadline non-preemptive task set $\tau^j$ is schedulable on a uniprocessor platform, if (and only if) Eq. (5) holds and Eq. (6) holds for every $\min_{\tau_i^j \in \tau^j} T_i^j \le t \le \max_{\tau_i^j \in \tau^j} T_i^j$.

$$\sum_{\tau_i^j \in \tau^j} RC_i^j / T_i^j \le 1.0, \tag{5}$$

$$\max_{\tau_i^j \in \tau^j | t < T_i^j} (RC_i^j - 1) + \sum_{\tau_i^j \in \tau^j} \left\lfloor \frac{t}{T_i^j} \right\rfloor \cdot RC_i^j \le t. \tag{6}$$

Note that if $\{\tau_i^j \in \tau^j | t < T_i^j\} = \emptyset$, then $\max_{\tau_i^j \in \tau^j | t<T_i^j}(RC_i^j - 1) = 0$.

If we change any $RC_i^j$, it may affect schedulability of all tasks in the subsystem $\mathcal{S}^j$. Thereby, for every change of $RC_i^j$, we conduct the schedulability test with Eqs. (5) and (6).

For another issue, the basic principle of minimizing battery aging is to minimize the variance of current load changes. Since the current load depends on the runtime power consumption of tasks, the $RC$ assignment should provide the runtime execution scheduler with as many as possible chances to adjust the current load. To this end, it should 1) assign more adjustable spaces (i.e., $RC_i^j - C_i^j$) to the task with higher power demand (i.e., $P_i^j$) and 2) distribute adjustable spaces to tasks as evenly as possible. This is because, the higher the power demand, the higher the impact on the current load; in addition, the even distribution is effective to maximize the sum of adjustable spaces subject to satisfying Eq. (6). Note that assigning a skewed $RC_i^j$ results in a rapid increase in $\max_{\tau_i^j \in \tau^j | t<T_i^j}(RC_i^j - 1)$ term in the left-hand side of Eq. (6).

**Algorithm 2** PR: reserved execution time ($RC$) assignment

For each $\mathcal{S}^j \in \mathcal{S}$,

1: For every $\tau_i^j \in \tau^j$, $RC_i^j \leftarrow C_i^j$
2: $\Phi \leftarrow$ an empty FIFO queue
3: Push every $\tau_i^j \in \tau^j$ into $\Phi$ in descending order of $P_i^j$
4: **while** $\Phi \neq \emptyset$ **do**
5: $\quad \tau_i^j \leftarrow$ pop a task from the head of $\Phi$
6: $\quad$ Increase $RC_i^j$ of $\tau_i^j$ by 1
7: $\quad$ **if** sched_test($\tau^j$) **then**
8: $\quad\quad$ Push $\tau_i^j$ into the tail of $\Phi$
9: $\quad$ **else**
10: $\quad\quad$ Decrease $RC_i^j$ of $\tau_i^j$ by 1
11: $\quad\quad$ **continue**
12: $\quad$ **end if**
13: **end while**

Here, we propose $RC$ assignment algorithm as shown in Algorithm 2. It increases $RC_i^j$ of every task $\tau_i^j$ in a round-robin manner (in descending order of $P_i^j$), subject to the task set is schedulable. To do this, the algorithm first sets every $RC_i^j$ as $C_i^j$ (Line 1), and pushes every $\tau_i^j$ into a FIFO queue ($\Phi$) in descending order of $P_i^j$ (Lines 2-3). After that, it increases $RC_i^j$ of the task $\tau_i^j$ popped from the head of $\Phi$ (Lines 5-6) and checks the schedulability of the task set $\tau^j$ with the increased $RC_i^j$ (Lines 7-13). Note that sched_test($\{\tau^j\}$) returns true if the task set $\tau^j$ satisfies Eqs. (5) and (6). If $\tau^j$ passes the test then it pushes $\tau_i^j$ into $\Phi$ again. Otherwise, it decreases $RC_i^j$ and excludes $\tau_i^j$ from $\Phi$ (i.e., not to push into $\Phi$). It repeats those routines until $\Phi$ becomes empty.

This algorithm exhibits a time complexity of $O(\max(T_i^j - C_i^j) \cdot st(\tau^j))$ for each subsystem $\mathcal{S}^j \in \mathcal{S}$, where $st(\tau^j)$ denotes the complexity of the schedulability test for $\tau^j$. One may think the complexity is high; however, it can be relaxed through increasing $RC_i^j$ by more than 1 or assigning multiple $RC$ values at the same time. Also, note that this algorithm runs offline.

### B. Execution Scheduling Part

Now, we present a scheduling algorithm used for the execution scheduling part (i.e., the PE policy) of the *RET* framework. This determines the execution start time of each reserved request of tasks. Again, the basic principle of minimizing battery aging is to minimize the variance of current load changes; moreover, this can be achieved by minimizing the sum of $I(t)^2$ (refer to Section III-A). One may find the minimum sum through a brute force approach, but it entails heavy computation, which is not affordable for the runtime scheduler. To calculate $I(t)^2$ of every combination of the execution start time, it takes up to $\prod_{j=1}^{n} \max_{\tau_i^j \in \tau^j}(RC_i^j - C_i^j)$ times of computation. Note that the execution start time $ts_i^j$ can be assigned in between $[t, t + RC_i^j - C_i^j)$.

Instead, we propose a greedy heuristic which finds the local-minimum sum of $I(t)^2$ with an affordable computation cost. To this end, our heuristic adopts two basic principles: 1) one by one assignment and 2) an interval sum of $I(t)$ minimization. Following the first principle, our heuristic determines $ts_i^j$ in turn from a task with the highest $P_i^j$. This strategy is effective to not only reduce the computation complexity but also minimize the sum of $I(t)^2$. In addition, according to the second principle, our heuristic finds $ts_i^j$ assignment which minimizes an interval sum of $I(t)$, instead of the sum of $I(t)^2$. This helps to avoid an expensive operation cost for multiplication while yielding a $ts_i^j$ assignment result which is equivalent to the minimum sum of $I(t)^2$. This is because of the following reason. Suppose that the current load $c$ is added to $I(t)$ at a time instant of $t$; the variance of $I(t)$ is then changed to $2I(t) * c - c^2$. Since $c^2$ is fixed regardless of $t$, minimizing the interval sum of $I(t)$ yields the minimum variance of $I(t)$; then, the minimum variance yields the minimum sum of $I(t)^2$, as mentioned in Section III-A.

**Algorithm 3** PE: execution start time assignment ($I(t)$)

$I(t)$: system-wide current load of executing tasks over time
*In $\mathcal{S}^j \in \mathcal{S}$, at a given current time $t_{cur}$,*

1: $\tau_i^j \leftarrow$ a reserved but not executed task in $VQ^j$
2: $minSum \leftarrow$ a maximum integer value
3: **for** $x$ in $[t_{cur}, rc_i^j - C_i^j)$ **do**
4: $\quad intSum \leftarrow \sum_{t=x}^{x+C_i^j} I(t) + P_i^j$
5: $\quad$ **if** $intSum < minSum$ **then**
6: $\quad\quad minSum \leftarrow intSum$
7: $\quad\quad ts_i^j \leftarrow x$
8: $\quad$ **end if**
9: **end for**
10: update $I(t)$ as if $\tau_i^j$ executes in $[ts_i^j, ts_i^j + C_i^j)$

Algorithm 3 describes our heuristic. A subsystem that has the reserved task $\tau_i^j$ (but not executed yet) with the highest $P_i^j$ runs the algorithm first. After that, other subsystems run the algorithm in turn, in descending order of $P_i^j$ of their reserved task $\tau_i^j$. It calculates the sum of $I(t)$ for the interval for which $\tau_i^j$ may execute (i.e., $[x, x + C_i^j)$), while changing $x$ in $[t_{cur}, rc_i^j - C_i^j)$ (Lines 3-4). It then assigns $ts_i^j$ as $x$ which yields the minimum interval sum of $I(t)$ (Lines 5-8). Once it

determines $ts_i^j$, it then updates the system-wide current load $I(t)$ as if $\tau_i^j$ will execute from $ts_i^j$, to allow other systems access to the up-to-date system-wide information.

This greedy heuristic effectively relaxes the time complexity of the execution start time assignment. It exhibits a complexity of $O(\max_{\tau_i^j \in \tau^j}(RC_i^j - C_i^j) \cdot n)$. In addition, it replaces expensive multiplication operations (i.e., the sum of $I(t)^2$) with the light-weight integer sum operations.

## VII. Evaluation

In this section, we present the evaluation of the proposed *RET* framework with the PR and PE algorithms. We first describe our simulation setup and provide simulation results in terms of current load variance and capacity degradation.

### A. Simulation Setup

*1) Task generation:* We generated task sets using UUniFast-Discard algorithm [24], which has been widely used in real-time systems studies with three parameters, utilization $U$ (0.25, 0.375, 0.5, 0.625, 0.75 and 0.875)[5], a number of subsystems $n$ (2, 4, 6, and 8), and a number of tasks $k$ (5, 10, 15, and 20).

For each subsystem, we generate a set of tasks such that their total utilization is $U$. The period $T_i^j$ of each task $\tau_i^j$ was chosen randomly from 10ms to 1000ms, while the quantum time unit was set to 10ms. As for power consumption, task generation is targeted at UAVs that commonly discharge at approximately 4C current loads, where 1C refers to the current rate when discharging the capacity in one hour. The current load $P_i^j$ of each task $\tau_i^j$ was selected randomly between $0.01$C and $\frac{4}{U \cdot n}$C. Taking the current load of UAVs into account, we use $\frac{4}{U \cdot n}$C as an upper bound such that the generated systems have an average current load of 4C. For fair simulation across different cases, we also scale current load use patterns to have 4C current load on average. For each simulation case, we generated 100 task sets, each of which is deemed schedulable according to the schedulability analysis of non-preemptive uniprocessor EDF scheduling [12], [13].

*2) Simulation:* For each simulation case, we perform the simulation of task scheduling for 1,000,000 time units, which produces the current load pattern of 10,000 seconds (recall one time unit corresponds to $10ms$). During the simulation, we run the following four scheduling algorithms:

- `Vanilla EDF` (baseline): In each subsystem, an execution request is selected by the EDF algorithm (prioritized by its absolute deadline). Once the execution request is selected, it starts its execution immediately and performs its execution during $C_i^j$ time units.
- `RET EDF`: In each subsystem, an execution request is selected by EDF. Once the execution request is selected, it starts its reservation during $RC_i^j$ time units (which is determined by `PR`). Each execution request starts its actual execution as soon as its reservation starts.
- `RET MIN`: In each subsystem, an execution request is selected by EDF. Once the execution request is selected, it starts its reservation during $RC_i^j$ time units (which is determined by `PR`). The actual execution of each execution request within its reservation is determined by a greedy algorithm that aims at minimizing peak power consumption, whose structure is similar to `PE`.
- `RET BAT`: In each subsystem, an execution request is selected by EDF. Once the execution request is selected, it starts its reservation during $RC_i^j$ time units (which is determined by `PR`). The actual execution of each execution request within its reservation is determined by `PE`.

We have two evaluation metrics: the variance of the current load and the lifespan of the battery. The first metric is for evaluating how well each scheduling algorithm follows the proposed scheduling principle, which is to minimize the variance of the current load, and the second metric is for translating scheduling performance to the impact of scheduling on battery aging. To measure the lifespan of the battery, we used our accurate battery aging simulator, which is described in Section III. The simulator calculates the lifespan of the battery as the time until the battery is considered as dead, i.e., until the battery loses 20% of its capacity. In discharge and charge cycles, batteries were to continuously and repeatedly follow given current load patterns (made by the task scheduling) until fully discharged, and then charged with 1C constant current. We note that we measured the variance of the current load for every task set generated for simulation, but we measured the lifespan of the battery for 10 task sets for each simulation case due to long simulation time; each battery simulation takes 10 to 40 hours in wall clock time.

### B. Simulation Results

*1) Minimizing variance approach:* Section III introduced the scheduling principle of minimizing the variance of current load for minimizing capacity degradation and thereby increasing lifespan. To evaluate this, we compared variance and lifespan, and Figure 5 shows their linear fit. Linear regression fitting revealed a negative correlation between the variance of current and battery lifespan. The result shows the R-squared value of 0.9016, which means there exists a strong relationship

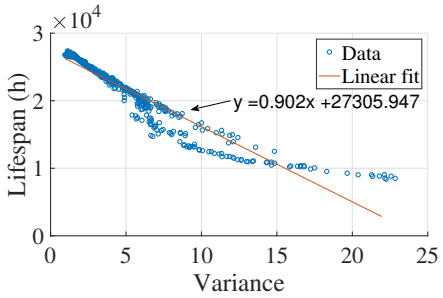
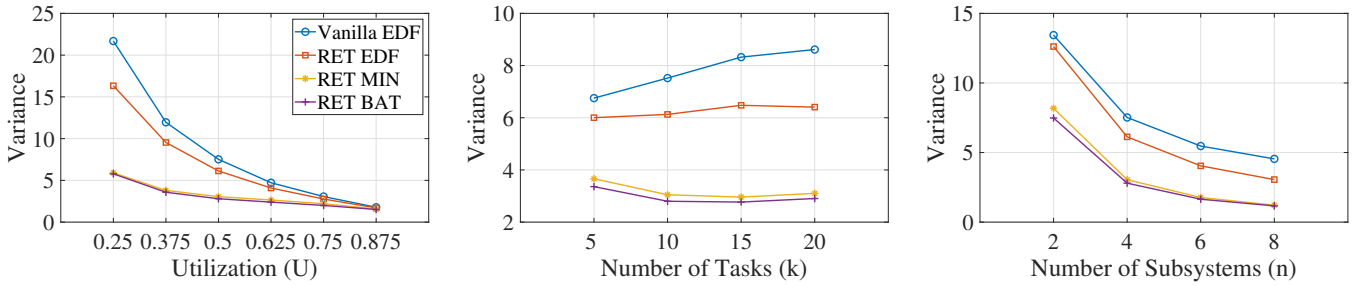


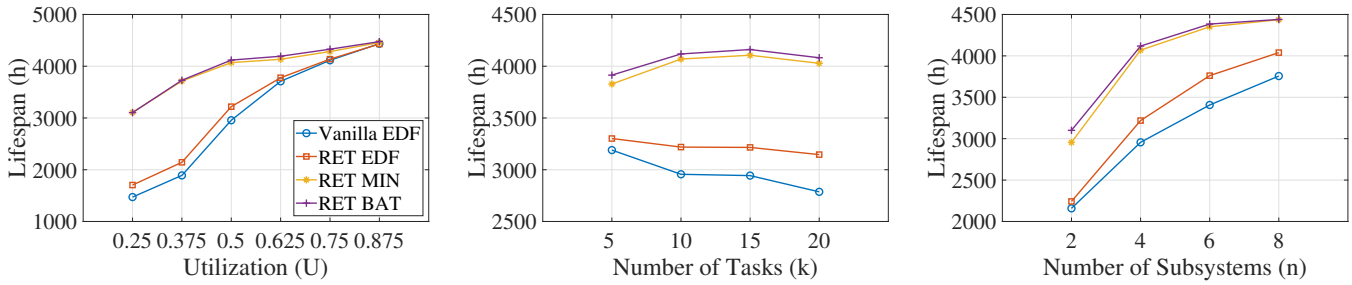

Fig. 5: Strong relationship between current variance and battery lifespan

[5]We excluded task sets with utilization of 0.125, because this setup yields a discharge current rate larger than the battery can support.




(a) Varying $U$ with $k = 10$ and $n = 4$ (b) Varying $k$ with $U = 0.5$ and $n = 4$ (c) Varying $n$ with $U = 0.5$ and $k = 10$

Fig. 6: Current load variance




(a) Varying $U$ with $k = 10$ and $n = 4$ (b) Varying $k$ with $U = 0.5$ and $n = 4$ (c) Varying $n$ with $U = 0.5$ and $k = 10$

Fig. 7: Battery lifespan

between current variance and battery lifespan. Also, this tendency continues within diverse condition changes, as shown in Figures 6 and 7. In conclusion, battery aging simulation testified that minimizing variance can slow down battery aging.

*2) RET framework:* In order to show the *RET* framework is effective for various system environments, we evaluated our framework with varying the number of tasks, system utilization, and the number of subsystems. As presented in Section VII-A, we varied simulation parameters. Figures 6 and 7 show the current load variance and the battery lifespan, respectively. In those figures, we confirm that the variance of current load directly affects the battery aging, regardless of the system setup.

Figures 6 and 7 show that `RET BAT` dominates all other approaches. Compared to the baseline (i.e., `Vanilla EDF`), `RET BAT` reduces the variance of current load by 56.57% and extends the battery lifespan by an average of 39.47%. In particular, when systems have larger $k$ and smaller $U$, the *RET* framework shows better performance. One interesting point is that when it comes to low utilization system ($u = 0.25$), `RET BAT` extends battery lifespan by up to 144.43% compared to the baseline. This is because, as the utilization decreases, *RET* assigns larger $RC_i^j$ to individual tasks, which provides more chances to reduce the variance of the current load at runtime. In addition, as the number of tasks increases, *RET* can distribute as evenly as possible and maximize the assigned $RC$ spaces to tasks. It is worthwhile noting that `RET MIN` (i.e., peak power minimization on top of the *RET* framework) results in a significantly longer battery lifespan compared to the baseline. This is because, *RET* provides rooms for reducing peak power, and this approach is also effective to reduce the variance of the current load. Beyond `RET MIN`, our approach `RET BAT` can most efficiently utilize assigned $RC$ spaces than other approaches. Thereby, `RET BAT` can minimize the variance of current load at runtime and achieve the longer battery lifespan.

*3) Discharge rates:* We perform additional simulations to see the impact of different discharge rates on battery lifespan. For the simulation case of $U$=0.5, $k$=10, and $n$=4, we revisited ten current load patterns and scaled their average discharge rate to 1C, 2C, and 4C. Figure 8 shows simulation results, indicating that the lifespan of battery rapidly decreases as the discharge rate increases. The figure shows that `RET BAT` improves lifespan compared to `Vanilla EDF` in 1C, 2C, 3C, and 4C discharge cases, extending 3.66%, 9.80%, 18.94%, and 39.33% on average and up to 4.78%, 12.57%, 24.40% and 80.40%, respectively. When the power load on the battery is high, that is, the discharge rate is large, a battery more rapidly becomes worn out, thus causing huge costs for maintaining the power system. The simulation results indicate that our method works even better in the situation.

*4) RC assignment:* We now evaluate our proposed $RC$ assignment algorithm in comparison with other algorithms as follows:

- `L`: High laxity ($T_i^j - C_i^j$) task prioritization, which tries to maximize the sum of $RC_i^j$.
- `P`: High $P_i^j$ task prioritization, which seeks to assign a larger $RC_i^j$ to a task with higher power consumption $P_i^j$.

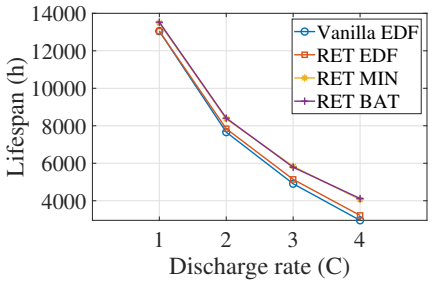



Fig. 8: Battery lifespan result, varying discharge rate with $U = 0.5$, $k = 10$ and $n = 4$

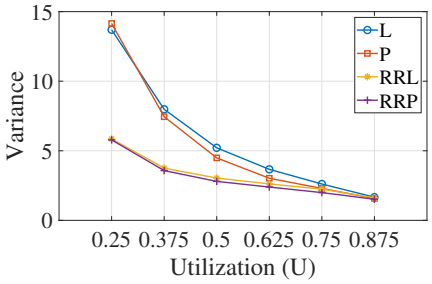



Fig. 9: Current load variance across different RC assignments

- `RRL`: Round-robin and tie break with high laxity ($T_i^j - C_i^j$) task prioritization, which seeks to distribute $RC_i^j$ evenly and increase the sum of $RC_i^j$.
- `RRP`: Round-robin and tie break with high $P_i^j$ task prioritization, which is our $RC_i^j$ assignment method.

We conducted simulations with the following parameters: $U$=(0.25, 0.5, and 0.75), $k$=10, and $n$=4. For each simulation case, we generated 100 task sets. Figure 9 depicts the average current load variance, and it shows that our assignment `RRP` always outperforms other policies. In particular, `RRP` results in significant improvement when task utilization is low ($U = 0.25$). This is because `RRP` effectively generates reserved execution times favorable to battery aging.

## VIII. Related Work

**Battery aging deceleration.** Reducing the capacity degradation of a battery is an important research field for the design of efficient and reliable power systems. There have been many studies to minimize battery aging by managing their charge patterns [25]–[28] or cooling/heating systems [29], [30], or redesigning their structures against aging [31], [32]. Some researchers made use of scheduling to manage the aging of a battery for some specific target systems [33], [34]. Nevertheless, there were few attempts to optimize battery usage subject to supporting real-time guarantees.

**Power-aware real-time scheduling.** Since power-consumption directly affects the performance of real-time embedded systems including a battery operation time, some researches have proposed power-aware real-time scheduling techniques. One of the major points of interest is managing peak-power consumption of real-time systems. For instance, Lee *et al.* [6] and Facchinetti *et al.* [7] have proposed real-time scheduling techniques to reduce the peak power consumption of processors and to maintain the system peak power consumption lower than a certain level, respectively. Another direction is to manage the overall power supply/consumption. Kim *et al.* [9] introduced a power scheduling framework to guarantee supplies of power load, by efficiently managing supplementary power sources such as solar panels in conjunction with a main Li-ion battery pack. For optimizing energy consumption, Kong *et al.* [8] and Castaings *et al.* [11] proposed resource management techniques that control processor frequency/voltage and exploit supercapacitors, respectively. Although such approaches consider power consumption factors which affect battery aging, they cannot achieve minimizing battery aging. This is because optimizing peak/overall power consumption does not necessarily follow the scheduling principle for less battery aging.

**Thermal-aware real-time scheduling.** Although we checked the temperature has a great influence on battery aging, existing studies for thermal-aware real-time scheduling were not highly relevant to battery aging. For example, Fisher *et al*. [35], Ahmed *et al*. [36], and Lee *et al*. [37] investigated thermal-aware scheduling on real-time systems while considering the thermal dynamics of processors. Although thermal factors directly affect battery aging, thermal-aware scheduling approaches are not suitable for minimizing battery aging. This is because they typically focus on thermal dynamics of subsystems (e.g., processors), not a battery itself; note that heat-generation of subsystems does not directly affect the temperature of the battery. Thereby, this approach is not suitable for the battery aging problem.

**Battery-aware real-time scheduling.** Several real-time scheduling techniques considering battery operations have been proposed. Luo *et el*. [10] proposed a DAG task scheduling algorithm that can improve the operation time of a battery, subject to real-time constraints. Lipskoch *et al*. studied to improve a battery operation time, through minimizing power-consumption with DVS (Dynamic Voltage Scaling) and the low-power mode of processors while considering the event stream model [38]. Although such researches proposed battery-aware real-time scheduling approaches, they mainly focused on improving the operation time of a single discharge cycle of a battery, which is known as not suitable for minimizing battery aging. In contrast, our scheduling framework directly considers minimizing battery aging based on a comprehensive understanding of battery dynamics, without compromising any timing constraints of real-time systems.

In another direction, Lipskoch *et al*. also studied an event stream based battery-aware energy feasibility analysis [39]; they proposed an analysis technique to guarantee power supplies to be higher than the power demand. However, this direction is not highly relevant to battery aging.

## IX. Discussion

In this section, we first discuss how *RET* works if the actual (as opposed to "worst") execution time of each execution request for each task $\tau_i^j$ is less than its WCET $C_i^j$. We then present how *RET* can be incorporated into existing power- or battery-related techniques.

### A. Consideration of actual execution time in real-world applications

For the comprehensive explanation, this paper assumed every execution request executes exactly as long as its WCET (i.e., $C_i^j$). However, many real-world tasks execute during less than its WCET, which can be dealt with the *RET* framework. That is, the *RET* framework runs the PE algorithm again at the completion of each execution request in order to recalibrate the start time of actual execution of every task $\tau_i^j$ that has reserved its subsystem but has not yet started its actual execution. To validate the *RET* framework in improving battery aging performance on the situation with actual execution time less than WCET, we conducted additional battery aging simulations for four scenarios, as follows.

- Original: all execution requests execute for exactly its WCET,
- Half: all execution requests execute for exactly its WCET/2,
- Uniform: all execution requests execute for execution time uniformly distributed in [1,WCET], and
- Normal: all execution requests execute for execution time determined by normal distribution in [1,WCET].

For the simulation setting of $U$=0.5, $k$=10, $n$=4, and discharge rate=4C, we revisited ten current load patterns as the previous evaluation. The result shows that regardless of early execution completion, the *RET* framework shows better battery aging performance compared to any other methods as plotted in Figure 10.

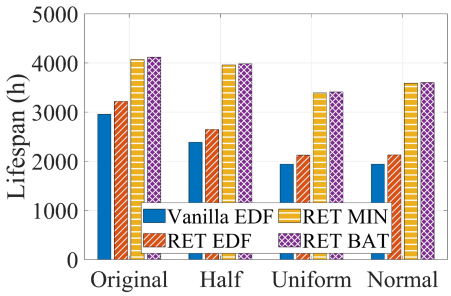



Fig. 10: Battery lifespan across different execution patterns

### B. Exploiting existing power- or battery-related techniques

While the *RET* framework decelerates battery aging by scheduling when each task is executed, one may wonder how the framework incorporates existing power- or battery-related techniques such as DVFS (Dynamic Voltage and Frequency Scaling) and cell balancing [8], [35], [38], which respectively reduces energy/power consumption and extends the operational time of the battery. In fact, the *RET* framework can employ those techniques in that there exists room for the framework to exploit those techniques. For example, once WCET is changed by DVFS, the *RET* framework can decrease/increase the reserved execution time, yielding different battery aging. On the other hand, it is possible that DVFS is applied *after* the *RET* framework determines the execution schedule. In the future, we would like to develop details on how to effectively employ the *RET* framework and the techniques *together*.

## X. Conclusion

This paper aims at minimizing the battery aging for battery-operated real-time systems. To this end, this paper introduces a task scheduling principle of minimizing the variance of total power consumption, which is essential for battery aging minimization. Furthermore, this paper proposes a new scheduling framework, *RET*, which allows addressing the problems of providing real-time guarantees *and* minimizing battery aging, in a sequential manner. This way, *RET* not only significantly reduces the complexity of achieving the goal but also enables to use existing scheduling techniques. Our extensive evaluations show that the proposed framework can extend the battery lifespan by up to 144.4%.

Although our target system consists of subsystems that accommodate non-preemptive power-consuming tasks and allow only one task to be executed at a time, the *RET* framework can be generally applicable to other types of tasks/subsystems such as preemptive tasks and/or subsystems that allows more than one task to be executed at the same time. In order for the *RET* framework to fully utilize existing scheduling algorithms and schedulability tests for other types of tasks/subsystems, it requires to tailor the reservation assignment part as well as the execution scheduling part; we leave it as future work.


## Acknowledgment

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2019R1A2B5B02001794, 2017H1D8A2031628). CPSlab ACK 추가 필요


## References


[1] C. Pillot, "The rechargeable battery market and main trends 2014–2025," in *31st International Battery Seminar & Exhibit*, 2015.

[2] F. Marra, E. Larsen, and C. Træholt, "Electric vehicles integration in the electric power system with intermittent energy sources-the charge/discharge infrastructure," 2013.

[3] J. Lee, E. Kim, and K. G. Shin, "Design and management of satellite power systems," in *2013 IEEE 34th Real-Time Systems Symposium*. IEEE, 2013, pp. 97–106.

[4] R. Alexander, "Analysis of aircraft power systems, including system modeling and energy optimization, with predictions of future aircraft development," Ph.D. dissertation, The Ohio State University, 2018.

[5] E. Ciancetta, M. Cimino, G. Cuzzocrea, G. Gervasio, E. Maiorano, I. Martinez, and L. Sanchez, "Electrical Power Subsystem for the Euclid Spacecraft," *E3S Web of Conferences*, vol. 16, no. 1, p. 18005, 2017.

[6] J. Lee, B. Yun, and K. G. Shin, "Reducing peak power consumption in multi-core systems without violatingreal-time constraints," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 4, pp. 1024–1033, 2013.

[7] T. Facchinetti and M. L. Della Vedova, "Real-time modeling for direct load control in cyber-physical power systems," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 689–698, 2011.

[8] F. Kong, W. Yi, and Q. Deng, "Energy-efficient scheduling of real-time tasks on cluster-based multicores," in *2011 Design, Automation & Test in Europe*. IEEE, 2011, pp. 1–6.

[9] E. Kim, J. Lee, L. He, Y. Lee, and K. G. Shin, "Offline guarantee and online management of power demand and supply in cyber-physical systems," in *2016 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2016, pp. 89–98.

[10] J. Luo and N. K. Jha, "Battery-aware static scheduling for distributed real-time embedded systems," in *Proceedings of the 38th annual Design Automation Conference*. ACM, 2001, pp. 444–449.

[11] A. Castaings, W. Lhomme, R. Trigui, and A. Bouscayrol, "Comparison of energy management strategies of a battery/supercapacitors system for electric vehicle under real-time constraints," *Applied Energy*, vol. 163, pp. 190–200, 2016.

[12] L. George, N. Rivierre, and M. Spuri, "Preemptive and non-preemptive real-time uniprocessor scheduling," Ph.D. dissertation, Inria, 1996.

[13] K. Jeffay, D. Stanat, and C. Martel, "On non-preemptive scheduling of periodic and sporadic tasks. 12th ieee real-time systems symposium," *San Antonio, TX*, 1991.

[14] X. Jin, A. Vora, V. Hoshing, T. Saha, G. Shaver, R. E. García, O. Wasynczuk, and S. Varigonda, "Physically-based reduced-order capacity loss model for graphite anodes in li-ion battery cells," *Journal of Power Sources*, vol. 342, pp. 750–761, 2017.

[15] L. Zhang, C. Lyu, G. Hinds, L. Wang, W. Luo, J. Zheng, and K. Ma, "Parameter sensitivity analysis of cylindrical lifepo4 battery performance using multi-physics modeling," *Journal of The Electrochemical Society*, vol. 161, no. 5, pp. A762–A776, 2014.

[16] R. Wright, C. Motloch, J. Belt, J. Christophersen, C. Ho, R. Richardson, I. Bloom, S. Jones, V. Battaglia, G. Henriksen, T. Unkelhaeuser, D. Ingersoll, H. Case, S. Rogers, and R. Sutula, "Calendar- and cycle-life studies of advanced technology development program generation 1 lithium-ion batteries," *Journal of Power Sources*, vol. 110, no. 2, pp. 445 – 470, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378775302002100

[17] M. Dubarry, N. Vuillaume, and B. Y. Liaw, "From single cell model to battery pack simulation for li-ion batteries," *Journal of Power Sources*, vol. 186, no. 2, pp. 500–507, 2009.

[18] A. Barré, B. Deguilhem, S. Grolleau, M. Gérard, F. Suard, and D. Riu, "A review on lithium-ion battery ageing mechanisms and estimations for automotive applications," *Journal of Power Sources*, vol. 241, pp. 680 – 689, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378775313008185

[19] H. J. Ploehn, P. Ramadass, and R. E. White, "Solvent diffusion model for aging of lithium-ion battery cells," *Journal of The Electrochemical Society*, vol. 151, no. 3, pp. A456–A462, 2004.

[20] A. M. Bizeray, S. Zhao, S. R. Duncan, and D. A. Howey, "Lithium-ion battery thermal-electrochemical model-based state estimation using orthogonal collocation and a modified extended Kalman filter," *Journal of Power Sources*, vol. 296, pp. 400–412, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.jpowsour.2015.07.019

[21] X. Zhang, "Thermal analysis of a cylindrical lithium-ion battery," *Electrochimica Acta*, vol. 56, no. 3, pp. 1246 – 1255, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0013468610014520

[22] K. Smith and C.-Y. Wang, "Power and thermal characterization of a lithium-ion battery pack for hybrid-electric vehicles," *Journal of Power Sources*, vol. 160, no. 1, pp. 662 – 673, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378775306001017

[23] A. M. Bizeray, J. Reniers, and D. A. Howey, "Spectral_li-ion_spm: Initial release," May 2016. [Online]. Available: https://doi.org/10.5281/zenodo.212178

[24] R. I. Davis and A. Burns, "Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems," in *2009 30th IEEE Real-Time Systems Symposium*. IEEE, 2009, pp. 398–409.

[25] L. He, Y.-C. Tung, and K. G. Shin, "icharge: User-interactive charging of mobile devices," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2017, pp. 413–426.

[26] S. Bashash, S. J. Moura, J. C. Forman, and H. K. Fathy, "Plug-in hybrid electric vehicle charge pattern optimization for energy cost and battery longevity," *Journal of power sources*, vol. 196, no. 1, pp. 541–549, 2011.

[27] A. Hoke, A. Brissette, K. Smith, A. Pratt, and D. Maksimovic, "Accounting for lithium-ion battery degradation in electric vehicle charging optimization," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 2, no. 3, pp. 691–700, 2014.

[28] L. He, E. Kim, and K. G. Shin, "*-aware charging of lithium-ion battery cells," in *Proceedings of the 7th International Conference on Cyber-Physical Systems*. IEEE Press, 2016, p. 26.

[29] E. Kim, K. G. Shin, and J. Lee, "Real-time battery thermal management for electric vehicles," in *ICCPS'14: ACM/IEEE 5th International Conference on Cyber-Physical Systems (with CPS Week 2014)*. IEEE Computer Society, 2014, pp. 72–83.

[30] G. Karimi and X. Li, "Thermal management of lithium-ion batteries for electric vehicles," *International Journal of Energy Research*, vol. 37, no. 1, pp. 13–24, 2013.

[31] A. Badam, R. Chandra, J. Dutra, A. Ferrese, S. Hodges, P. Hu, J. Meinershagen, T. Moscibroda, B. Priyantha, and E. Skiani, "Software defined batteries," in *Proceedings of the 25th Symposium on Operating Systems Principles*. ACM, 2015, pp. 215–229.

[32] S. Rothgang, T. Baumhöfer, H. van Hoek, T. Lange, R. W. De Doncker, and D. U. Sauer, "Modular battery design for reliable, flexible and multi-technology energy storage systems," *Applied Energy*, vol. 137, pp. 931–937, 2015.

[33] B. Zhou, X. Liu, Y. Cao, C. Li, C. Y. Chung, and K. W. Chan, "Optimal scheduling of virtual power plant with battery degradation cost," *IET Generation, Transmission & Distribution*, vol. 10, no. 3, pp. 712–725, 2016.

[34] L. Liu, H. Sun, C. Li, T. Li, J. Xin, and N. Zheng, "Managing battery aging for high energy availability in green datacenters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 12, pp. 3521–3536, 2017.

[35] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele, "Thermal-aware global real-time scheduling on multicore systems," in *2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2009, pp. 131–140.

[36] R. Ahmed, P. Ramanathan, and K. K. Saluja, "Necessary and sufficient conditions for thermal schedulability of periodic real-time tasks under fluid scheduling model," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 15, no. 3, p. 49, 2016.

[37] Y. Lee, H. S. Chwa, K. G. Shin, and S. Wang, "Thermal-aware resource management for embedded real-time systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2857–2868, 2018.

[38] H. Lipskoch, "Optimisation of battery operating life considering software tasks and their timing behaviour," Ph.D. dissertation, Universität Oldenburg, 2010.

[39] H. Lipskoch, K. Albers, and F. Slomka, "Battery discharge aware energy feasibility analysis," in *Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*. ACM, 2006, pp. 22–27.