Maximum-Utility Scheduling of Operation Modes With Probabilistic Task Execution Times Under Energy Constraints

Wan Yeon Lee, Hyogon Kim, and Heejo Lee, *Member, IEEE*

***Abstract*—We propose a novel scheduling scheme that determines the instant operation modes of multiple tasks. The tasks have probabilistic execution times and are executed on discrete operation modes providing different utilities with different energy consumptions. We first design an optimal offline scheduling scheme that stochastically maximizes the cumulative utility of the tasks under energy constraints, at the cost of heavy computational overhead. Next, the optimal offline scheme is modified to an approximate online scheduling scheme. The online scheme has little runtime overhead and yields almost the maximum utility, with an energy budget that is given at runtime. The difference between the maximum utility and the output utility of the online scheme is bounded by a controllable input value. Extensive evaluation shows that the output utility of the online scheme approaches the maximum utility in most cases, and is much higher than that of existing methods by up to 50% of the largest utility difference among available operation modes.**

***Index Terms*—Approximate scheduling, energy constraint, maximum utility, optimal scheduling, probabilistic execution time.**

I. Introduction

FOR battery-operated mobile devices, energy management is an issue of paramount importance. Numerous studies [1]–[5] have been conducted to reduce the operational energy consumption of devices and thereby extend the battery lifetime while satisfying the required performance. However, there has been a dearth of works accommodating performance degradation for the purpose of completing the given task. In many tasks on mobile devices, such flexibility could be of considerable value. On mobile Internet Protocol Television (IPTV) devices, for instance, completing a low-quality broadcast of live sports is usually preferable to sudden termination of a high-quality broadcast due to lack of available energy [6]. Attempting to reduce the energy consumption without performance degradation, as in the aforementioned schemes, has a more limited solution space that can lead to invaluable loss of output quality, i.e., unexpected task abortion caused by depletion of battery energy. Thus, we are motivated to develop an energy management scheme that maximizes the system performance while sustaining the battery lifetime for the specified duration.

In the considered systems, tasks choose one of the *energy-aware operation modes* and can change to another operation mode at any time. Each of the energy-aware operation modes provides a different *utility* with corresponding energy consumption. Representative platforms of energy-aware operation modes are energy-aware networking [1], [7], adaptive speed control of motors [8], [9], and CPU voltage and frequency scaling [10], [11]. When the network card, the motor, and the CPU operate with higher energy consumption, they yield superior performance, i.e., transmission of larger images, faster motor revolutions, and faster CPU processing, respectively. In this paper, these performance metrics are quantified via *utility*. If tasks are executed in an energy-aware operation mode with a higher utility, the battery is consumed at a faster rate and vice versa. In this setting, we investigate how to *maximize the cumulative utility* obtained from tasks under the constraint of sustaining the energy supply until their completion.

The completion times of most multimedia tasks such as streaming services of IPTV [6] and search and rescue of mobile robots [12] are usually not predictable. This uncertainty makes it difficult to specify the execution time of tasks and determine the optimal energy consumption rate (i.e., the energy-aware operation mode). The most conservative model for dealing with the unpredictability of the execution time is to adopt the worst-case execution time. However, this approach may result in severely inefficient utilization of resources, because the worst-case execution time is usually set to be sufficiently large to support unexpectedly long execution cases. As an alternative, our method exploits the probabilistic execution time to determine an appropriate energy-aware operation mode at each instant. The proposed method assigns higher utility higher energy operation mode to those parts of the task that are more likely to be executed.

As an application example of the proposed idea, consider a Major League Baseball game played on a mobile IPTV device. The playing times of baseball games are between 2 and 4 h in most cases, but rarely exceed four h with the worst-case time of about 8 h. Then, the estimated probability of continuing a game is 100% for 2 h, decreases beyond 2 h with time, and eventually

Manuscript received September 27, 2008; revised January 10, 2009, April 12, 2009, and June 18, 2009. Current version published September 18, 2009. This work was supported by the Ministry of Knowledge Economy (MKE), Korea, under the Information Technology Research Center (ITRC) support program supervised by the Institute for Information Technology Advancement (IITA) (IITA-2009-C1090-0904-0001). This paper was recommended by Associate Editor G. E. Martin.

W. Y. Lee is with the Department of Computer Engineering, Hallym University, Chunchon 200-702, Korea (e-mail: wanlee@hallym.ac.kr).

H. Kim and H. Lee are with the Division of Computer and Communication Engineering, Korea University, Seoul 136-701, Korea (e-mail: hyogon@korea.ac.kr; heejo@korea.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2009.2026352

falls to 0% at 8 h. Given a limited budget of battery energy, the proposed method assigns a high-resolution MPEG-4 streaming mode [2], [13] for 2 h, and gradually changes it to lower resolution modes with progressed running time. Even though the later processing parts provide lower resolution when being executed, they are rarely performed due to their low probability. This stochastic approach provides a larger accumulation of mean resolution obtained for the worst execution time than assigning a fixed median-resolution mode for the worst time.

Some previous studies [14]–[17] have considered the control mechanisms for sustaining the energy supply for a specified duration. They assigned a fixed operation mode whose total energy consumption for the specified duration is no larger than a given energy budget. However, these studies only considered the energy management of tasks with a fixed execution time, i.e., the worst-case execution time, without considering tasks with a wide range of execution times. A few studies on real-time scheduling [10], [11], [18]–[21] have considered the stochastic approach for exploiting the probability distribution of task execution times. However, these studies pursued a different goal on different environments, i.e., minimizing the energy consumption of CPU while meeting the deadlines of real-time tasks. In contrast, this paper addresses the problem of maximizing the total utility of multiple tasks while sustaining the energy supply until the worst execution time.

The contribution of this paper is twofold. The first contribution is to design an optimal scheduling scheme that theoretically solves the problem of maximizing the total mean utility of multiple tasks under energy constraints, where the tasks have probabilistic execution times and are executable on a set of discrete energy-aware operation modes. The derived optimal scheduling scheme is useful for offline processing, but impractical for online processing, due to its heavy computational overhead. Hence, as the second and main contribution, we propose an approximate online scheduling scheme that operates with little runtime overhead and yields almost the maximum utility. The approximate online scheme is implemented in a semistatic manner, where candidate solutions are prepared at compile time and one of them is selected at runtime in accordance with the given energy budget. The online scheme can adjust its output quality and runtime overhead with a controllable input value that sets a bound on the difference between the output utility and the maximum utility. Both the optimal offline scheme and the approximate online scheme guarantee that the minimum-energy supply will be sustained until the worst execution time without energy depletion.

Our extensive evaluation shows that the output utility of the online scheme approaches the maximum utility of the optimal offline scheme in most cases, and is much higher than that of the existing methods [14], [15] assigning the same operation mode for the worst execution time. In the best case, the online scheme increases the output utility of the existing methods by 50% of the largest utility difference among the available operation modes.

The rest of this paper is organized as follows. Section II provides the preliminary knowledge facilitating the understanding of the proposed method, and formulates the problem tackled in this paper. Section III describes the optimal offline scheme and Section IV describes the approximate online scheme in detail. Section V evaluates the proposed online scheme through extensive simulations. Section VI reviews the related prior work, and Section VII finally concludes this paper.

II. Preliminaries and Problem Formulation

A. Energy-Aware Operation Mode and Battery Lifetime

Not surprisingly, empirical studies [1], [5], [14], [22] show that component devices can run on disparate operation modes and yield better performance on the operation mode consuming higher energy. The power dissipation of wireless LAN cards is linearly proportional to the size of transmitted packets [22]; the power dissipation of an IEEE 802.11 PC Card is "1.9 $\times$ the size of transmitted packet $(\mu\mathrm{W}\cdot\mathrm{sec}/\mathrm{byte}^7) + 456\ (\mu\mathrm{W}\cdot\mathrm{sec})$" for point-to-point sending. The power dissipation of dynamic voltage and frequency scaling (DVFS)-enabled CPU is exponentially proportional to the clock frequency (processing speed) [10], [11]; the power dissipations of the XScale processor are 80 mW for 150 MHz, 400 mW for 600 MHz, and 1600 mW for 1000 MHz. The power consumption (energy consumption rate) of the operation modes regularly fluctuates around a fixed average.

Essentially, there is a tradeoff between the power consumption and the performance of device's operation modes such as network cards, motors, display screens, and memories. Fine-granular-scalability (FGS) video coding [2], [13] changes the size of transmitted images per second. Adaptive motor control [8], [9] scheme changes the moving speed of mobile robots. Adaptive forward error correction (FEC) [23] scheme changes the resiliency to errors. Power-aware backlight scaling [24] scheme changes the brightness of the display screen. Power-aware page allocation [25] and adaptive disk spin-down [26] schemes change the average access time of the memories.

In this paper, the aforementioned performances of component devices are associated with the notion of *utility*. We assume that the utility values corresponding to the operation modes of each device are inherently given. For the preference discrimination of disparate devices or tasks, weights are assigned to the utility values of their operation modes by the user or the system administrator. In addition, in order to make the model more realistic, we consider a finite set of discrete operation modes rather than infinite continuum of modes. When multiple component devices are manipulated by a task, combinations of the corresponding operation modes are handled as a set of available operation modes [27].

Several tools have been developed to allow the operating system (OS) to accurately measure the energy consumption rates for disparate operation modes, such as PowerScope [14], Advanced Power Management, Advanced Configuration and Power Interface, PCCextend 140 CardBus extender [22], and an application-level tool [28]. The lifetime prediction techniques [29], [30] enable OS to estimate the residual lifetime of a battery based on the energy consumption rate of running tasks and the available energy in the battery. To manage the residual battery lifetime for a given finite temporal horizon, OS can determine the energy consumption rate maximally allocatable to the execution of tasks [14], [15].

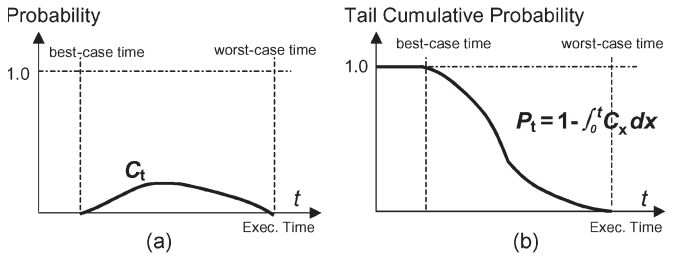

Fig. 1. (a) Probability distribution of task completion times. (b) Tail cumulative distribution of (a).

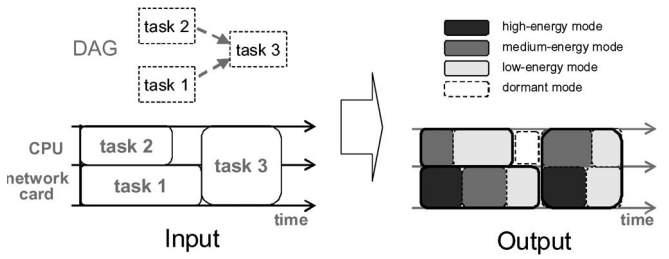

Fig. 2. Operating example of the proposed scheduler.

B. Probabilistic Execution Time

For simplicity and safety, battery-operated systems could be designed to schedule tasks according to their worst execution time, i.e., to evenly distribute available energy over the worst execution case. However, this would lead to unnecessarily low utility in the case where the execution time is shorter than the worst case. Indeed, tasks can have a wide range of execution times, due to the diversity of input data or disparate user requests about them [11], [31], [32]. On the other hand, task scheduling schemes tailored to the average execution times are likely to incur untimely energy depletion, in the case where the task completion time approaches the worst case.

To overcome this problem, we propose exploiting the probability distribution of task execution times. The distribution could be derived from statistical models of the variation sources, online profiling, or offline profiling [10], [11]. Fig. 1(a) shows a probability distribution example of task completion times, and Fig. 1(b) shows the tail cumulative distribution of the probability shown in Fig. 1(a). When the probability at time t is denoted as C_t in Fig. 1(a), its tail cumulative probability at time t is $(1 - \int_0^t C_x dx)$ in Fig. 1(b). Henceforth, we denote this tail cumulative probability at time t as P_t, where the task execution time is not affected by the operation modes.[1] In other words, P_t is the probability that the task is still running at time t. Note that $P_{t_1} \geq P_{t_2}$ for $t_1 < t_2$, because the cumulative distributions of all probability functions are nondecreasing and thus their tail distributions are nonincreasing.

C. System and Task Models

Given tasks with arrival times and execution order, the designed scheduler determines the instant operation mode of devices manipulated by each task. The given tasks may run concurrently, have a precedence of execution, or have different arrival times. Their exact execution times are unknown until completing the execution, but their probabilistic execution times are known in advance. The scheduler decides the instant operation modes from the task starting points. Fig. 2 shows an operating example of the scheduler, where task 1 and task 2 start their execution from their arrival times and task 3 starts its execution upon the completion of task 1. The tasks manipulate one or more component devices for their execution, and are executable on a finite set of operation modes that is the combination set of available operation modes of the manipulated devices. Each device works for one task at a time, and sequentially for multiple tasks. Based on the progressed running time, the scheduler changes the operation modes of manipulated devices that work independently from the operation modes of other devices.

In this paper, we do not consider arrivals of unexpected tasks and changing the operation mode of a *dependent* device that is affected by the operation modes of other devices. We also ignore the energy consumption of the *dormant* operation mode, where devices are deactivated, and the energy overhead required to activate and deactivate devices.[2] The following notations are used to denote the parameters of given tasks and available energy.

O_j^i	jth operation mode of the ith task.
$\mathcal{F}_r(O_j^i) = R_j^i$	Power consumption (energy consumption rate) of the operation mode O_j^i.
$\mathcal{F}_u(O_j^i) = U_j^i$	Utility of the operation mode O_j^i.
N^i	Number of operation modes available for the ith task.
π_n^i	Transition time from O_n^i to O_{n-1}^i.
P_t^i	Probability that the ith task is still running at time t.
W^i	Worst execution time of the ith task.
$E_{\max}$	Available energy budget.
E^i	Energy budget allocated to the ith task among $E_{\max}$.

If $\mathcal{F}_r(O_x^i) < \mathcal{F}_r(O_y^i)$ but $\mathcal{F}_u(O_x^i) \geq \mathcal{F}_u(O_y^i)$ for any x and y, the operation mode O_y^i is excluded due to its inefficiency. The available operation modes of each ith task are sorted in increasing order of their energy consumption rates (or utility values) and denoted as $O_1^i = [\mathcal{F}_r(O_1^i), \mathcal{F}_u(O_1^i)] = [R_1^i, U_1^i], \ldots, O_{N^i}^i = [\mathcal{F}_r(O_{N^i}^i), \mathcal{F}_u(O_{N^i}^i)] = [R_{N^i}^i, U_{N^i}^i]$. Time notations π_n^i, t, and W^i denote the elapsed times from the task starting points.

In the considered system, the information about P_t^i, W^i, and O_j^i of all tasks is given to the scheduler in advance. The value of $E_{\max}$, denoting the residual energy of a battery, is dynamically given at runtime. The considered system can change the running operation mode at any time, where the continuous time are

[1]Where the operation modes affect the task execution time but not the task computation amount; probabilistic computation amount is used instead of probabilistic execution time. For example, in DVFS operation modes, P_c is used to formulate the probability of the cth CPU computation cycle required for processing a task, where the total CPU computation cycles are fixed but the execution speed depends on the CPU clock frequency. In this paper, we exclude the operation mode that changes both the execution speed and the computation amount per unit time.

[2]The energy consumption of the dormant mode for the maximum duration is subtracted from the available energy budget, and energy consumptions of energy-aware operation modes are replaced with their energy differences from the dormant mode. In addition, a portion of available energy is reserved for activating and deactivating devices.

approximated with fine-grain discrete values. For convenience, we handle the fine-grain time values with natural numbers through unit conversion, e.g., 23.000001 s $\rightarrow$ 23 000 001 μs.

D. Problem Formulation of Task Scheduling

The problem handled in this paper is to determine the instant operation mode of M tasks with probabilistic execution times, to maximize the total cumulative mean utility while sustaining the energy supply until the worst execution time with a given energy $E_{\max}$. Energy consumption and output utility of a task depend only on its instant operation mode, but not on those of other tasks. The remaining issues are how to distribute the available energy to the M tasks and how to use the distributed energy as a function of t and P_t^i.

Let $\mathcal{SO}_t^i$ denote the selected operation mode that will run at time t for the ith task, where $\mathcal{SO}_t^i \in \{O_1^i, \ldots, O_{N^i}^i\}$. Then, the problem of maximizing the cumulative mean utility of a single task without energy depletion for the worst execution time can be formulated as follows:

$$\begin{aligned} \text{Maximize} \quad & \int_0^{W^i} \mathcal{F}_u\left(\mathcal{SO}_t^i\right) \cdot P_t^i dt \\ \text{subject to} \quad & \int_0^{W^i} \mathcal{F}_r\left(\mathcal{SO}_t^i\right) dt \le E_{\max}. \end{aligned} \tag{1}$$

The extended problem for the M tasks can be formulated as follows:

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{M} \int_0^{W^i} \mathcal{F}_u\left(\mathcal{SO}_t^i\right) \cdot P_t^i dt \\ \text{subject to} \quad & \sum_{i=1}^{M} \int_0^{W^i} \mathcal{F}_r\left(\mathcal{SO}_t^i\right) dt \le E_{\max}. \end{aligned} \tag{2}$$

A derived solution determines $\mathcal{SO}_t^i$ for each i and $0 \le t \le W^i$. In other words, it determines the starting and ending times of each operation mode.

To provide at least the minimum utility U_1 for the worst execution time, it is necessary to give more than a lower energy bound, i.e., $E_{\max} \ge \sum_{i=1}^{M} R_1^i \cdot W^i$. Henceforth, we consider the case of $E_{\max} \ge \sum_{i=1}^{M} R_1^i \cdot W^i$ and refer to a schedule producing the maximum utility as *Optimal Schedule*.

III. Optimal Offline Scheduling Scheme

In this section, we first find an Optimal Schedule of a single task and, next, exploit it to derive an Optimal Schedule of multiple tasks.

A. Optimal Schedule of a Single Task

The Optimal Schedule has the following properties, formally proved in Appendix.

Lemma 1: The Optimal Schedule excludes the operation mode O_y such that $(U_y - U_x)/(R_y - R_x) < (U_z - U_y)/(R_z - R_y)$ for any $R_x < R_y < R_z$ (and $U_x < U_y < U_z$).

Lemma 2: In an Optimal Schedule, $\mathcal{F}_u(\mathcal{SO}_{t_1}) \ge \mathcal{F}_u(\mathcal{SO}_{t_2})$ and $\mathcal{F}_r(\mathcal{SO}_{t_1}) \ge \mathcal{F}_r(\mathcal{SO}_{t_2})$ for any $0 < t_1 < t_2 \le W$.

By Lemma 1, we hereafter discard the operation modes whose utility values are not in accordance with the concave function of their energy consumption rates. Then, the utility is a concave function of the energy consumption rate, i.e., $(U_n - U_{n-1})/(R_n - R_{n-1}) \ge (U_{n+1} - U_n)/(R_{n+1} - R_n)$ for any $2 \le n \le (N-1)$. By Lemma 2, we consider only the transitions from high-utility operations to low-utility operations. Lemma 2 means that $0 \le \pi_n \le \pi_{n-1} \le W$ for any $2 \le n \le N$ in an Optimal Schedule, where π_n is the transition point from O_n to O_{n-1}. Then, (1) can be reformulated as follows:

$$\begin{aligned} \text{Maximize} \quad & U_N \cdot \int_0^{\pi_N} P_t dt + U_{N-1} \cdot \int_{\pi_N}^{\pi_{N-1}} P_t dt \\ & + \cdots + U_1 \cdot \int_{\pi_2}^{\pi_1} P_t dt \\ \text{subject to} \quad & R_N \cdot \pi_N + R_{N-1}^i \cdot (\pi_{N-1} - \pi_N) \\ & + \cdots + R_1^i \cdot (\pi_1 - \pi_2) \le E_{\max} \end{aligned} \tag{3}$$

where $\pi_1 = W$ to support at least the minimum utility U_1 for the worst time, and $\pi_n = 0$ or $\pi_n = \pi_{n+1}$ if O_n is not used.

If we know the values of all π_ns, we can directly obtain the Optimal Schedule. Unfortunately, however, the values of π_ns for $n \ge 2$ depend on the given $E_{\max}$. In order to find the values of π_ns, we first examine the relationship among the values of π_ns and, next, exploit it to obtain the values of π_ns associated with the value of $E_{\max}$. The following Theorem 1, proved in Appendix, verifies the relationship among the values of π_ns for $n \ge 2$.

Theorem 1: The values of π_ns in an Optimal Schedule obey the following relationship:

$$P_{\pi_x} \cdot \frac{U_x - U_{x-1}}{R_x - R_{x-1}} = P_{\pi_y} \cdot \frac{U_y - U_{y-1}}{R_y - R_{y-1}}$$

$2 \le x < y \le N$ such that $W \ge \pi_x > \pi_y > 0$.

From a fixed value π_2, we can calculate deterministic values of $\pi_3, \ldots, \pi_N$ satisfying the relationship in Theorem 1. Exhaustive searching of $\pi_3, \ldots, \pi_N$ from $\pi_2 = 1$ to $\pi_2 = W$ enables us to find the exact π_ns matching $E_{\max}$. The procedure for finding π_ns works as follows. Initially, it assigns $\pi_2 = 1$ and searches for the time point π_n such that

$$P_{\pi_n} = P_{\pi_2} \cdot \frac{U_2 - U_1}{R_2 - R_1} \cdot \frac{R_n - R_{n-1}}{U_n - U_{n-1}}, \quad \text{for } 3 \le n \le N.$$

Because $0 \le P_{\pi_n} \le 1$ by the definition of P_t, the value of π_n such that $P_{\pi_n} > 1$ is set to zero. It calculates the energy consumption of the searched schedule and compares it with the given energy $E_{\max}$. If the energy consumption of the searched schedule is smaller than the given energy, it increases the value of π_2 and again searches for the time point π_n for $n \ge 3$. If

$\pi_2 = W$ but the energy consumption is still smaller than the energy budget, it increases the value of π_3 after fixing π_2 to W and searches for the remaining time point π_n for $n \geq 4$. Similarly, if $\pi_j = W$ but the energy consumption is still smaller than the given energy, it increases the value of π_{j+1} instead of π_j. This process is repeated until the energy consumption of the searched schedule is equal to $E_{\max}$.

The computational time complexity of the aforementioned procedure is $O(N \cdot (\log_2 W)^2)$ in the average case. If the energy consumption when $\pi_2 = W$ is smaller than $E_{\max}$, the replacement of the base value π_2 with π_j such that $j > 2$ is repeated at most $(N-2)$ times until the energy consumption when $\pi_j = W$ is larger than $E_{\max}$. With a base value of π_j, the bisectional search operation of the remaining time point π_n satisfying the relationship in Theorem 1 requires $O(\log_2 W)$ steps for each $j < n \leq N$. When the base value of π_j is selected in a bisectional manner,[3] the selection operation of the fixed π_j is repeated at most $O(\log_2 W)$ times. Then, $O((N-2) \cdot \log_2 W + N \cdot \log_2 W \cdot \log_2 W) = O(N \cdot (\log_2 W)^2)$.

B. Optimal Schedule of Multiple Tasks

Given P_t^i, O_j^is, and W^i, the exhaustive search procedure described in Section III-A can derive the maximum of (3), which depends only on the distributed energy E^i. If we can obtain the maximum of (3) for any energy E^i, the result of (2) is entirely dominated by the decision on how to distribute the energy $E_{\max}$ to the M tasks. When Ψ^i denotes the maximum utility derived from (3) with energy E^i, (2) can be reformulated as follows:

$$\text{Maximize} \quad \sum_{i=1}^{M} \Psi^i \qquad \text{subject to} \quad \sum_{i=1}^{M} E^i \leq E_{\max}. \quad (4)$$

As the allocated energy E^i increases, the utility Ψ^i increases, but the utility increment per unit energy $\Psi^i/\partial E^i$ decreases by Lemma 3 in Appendix. In this case, we can apply the Lagrange Multiplier method [33] to obtain the solution to (4). Then

$$L(E^1, \ldots, E^N, \lambda) = \sum_{i=1}^{M} \Psi^i + \lambda \cdot \left(E_{\max} - \sum_{i=1}^{M} E^i \right)$$

$$\frac{\partial L}{\partial \lambda} = E_{\max} - \sum_{i=1}^{M} E^i = 0$$

$$\frac{\partial L}{\partial E^k} = \frac{\Psi^k}{\partial E^k} - \lambda = 0, \quad \text{for } 1 \leq k \leq N.$$

The aforementioned equations verify that the maximum of (4) occurs when $\sum_{i=1}^{M} E^i = E_{\max}$ and the values of $\Psi^i/\partial E^i$ for $1 \leq i \leq N$ are equal to the largest. In other words, (4) maximizes the total utility increments gained with the allocated energy E^is. The maximum of (4) is achieved by incrementally allocating an additional unit energy to the task providing the largest utility increment with the additional unit energy.

[3]A more specific procedure for searching for the values of πs in a bisectional manner is described in Section IV-A.

To provide at least the minimum utility until the worst execution time, $R^i \cdot W^i$ is initially assigned to each E^i. A numerical procedure can distribute $(E_{\max} - \sum R_1^i \cdot W^i)$ to the M tasks, to maximize the total utility increments obtained with the remaining energy $(E_{\max} - \sum R_1^i \cdot W^i)$. It calculates the utility increment of each task when allocating the remaining energy evenly to all tasks (calculating the utility increment of the additional energy $(E_{\max} - \sum_{i=1}^{M} E^i)/M$ for each task). Next, it selects the task having the largest utility increment and actually allocates the additional energy only to the task. This process is repeated until all the remaining energy is allocated to the M tasks. The computational complexity of the numerical procedure is $O(\log_{M/(M-1)} E_{\max} \cdot \sum_{i=1}^{M} \{N^i \cdot (\log_2 W^i)^2 + W^i\})$ in the average case. The operation to search for the values of π_ns with an increased energy requires $O(N^i \cdot (\log_2 W^i)^2)$ steps for each task, as explained in Section III-A. The operation to calculate the utility increment obtained with the additional energy requires $O(W^i)$ steps for each task. The operations to search for π_ns and calculate the utility increment are repeated $O(\log_{M/(M-1)} E_{\max})$ times, where $(1-(1/M))^{\alpha} \cdot (E_{\max} - \sum_{i=1}^{M} R_1^i \cdot W^i) \simeq 1$.

The aforementioned scheduling scheme to find an Optimal Schedule is the best for offline processing. However, when the available energy budget is dynamically given at runtime, its computational overhead is too heavy to be carried out at runtime because the worst execution time W^i is usually set to a sufficiently large value. Furthermore, it is preferable to check the residual battery energy periodically and update the solution frequently [14], [15], in order to minimize the potential loss caused by an error in measuring the residual battery energy. Overestimation of the residual energy prevents the task from completing its worst-case execution, and underestimation prevents the task from exploiting more energy resources. In the next section, therefore, we propose an online scheduling scheme that operates with acceptable runtime overhead and provides almost the maximum utility at the cost of a bounded utility decrease.

IV. Approximate Online Scheduling Scheme

The proposed online scheme prepares candidate solutions at compile time and selects one of the solutions at runtime. In Section IV-A, we explain how to prepare a set of candidate solutions at compile time. In Section IV-B, we explain how to select one of the solutions prepared for each task at runtime in accordance with a dynamically given energy budget. In Section IV-C, we discuss when to update the scheduling results with a newly measured energy budget and how to accommodate the overhead required to change operation modes.

A. Precomputation of Candidate Solutions at Compile Time

When discrete energy values increase by a given ratio, the online method finds and stores the corresponding maximum-utility schedules for the runtime selection. The precomputed schedules are referred to as *candidate solutions*. To determine how many candidate solutions are prepared at compile time, we define an input value of *Error Bound* and denote it as ϵ. When

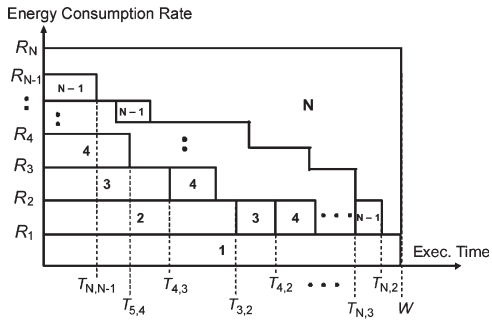

Fig. 3. $(N+1)$ solutions found by the coarse-grain partitioning procedure.

candidate solutions are sorted in increasing order of their energy consumptions and e_k^i denotes the energy consumption of the kth solution for the ith task, the method finds an intermediate solution until

$$\frac{e_{k+1}^i - e_k^i}{e_k^i} \leq \epsilon, \quad \text{for each } k \text{ and } i.$$

The value ϵ of Error Bound is given by the user or system administrator, in order to set a bound on the utility difference between the candidate solution selected at runtime and the Optimal Schedule derived with the energy given at runtime.

First, the online method finds $(N+1)$ candidate solutions that inherently use different subsets of operation modes, which are shown in Fig. 3. It is easy to find the candidate solution using only the operation mode O_1 with the maximal energy consumption, which is $\pi_1 = W$ and $\pi_n = 0$ for $n \geq 2$ and whose energy consumption is $e_1 = R_1 \cdot W$. To find the candidate solution using only O_2 and O_1 with the maximal energy consumption, it exploits the relationship of π_ns in Theorem 1. The value of π_2 is calculated with $\pi_3 = 0$ and $P_{\pi_3} = 1$ such that

$$P_{\pi_2} \cdot \frac{U_2 - U_1}{R_2 - R_1} = P_{\pi_3} \cdot \frac{U_3 - U_2}{R_3 - R_2} = \frac{U_3 - U_2}{R_3 - R_2}.$$

If $\pi_3 = 0, O_3, \ldots, O_N$ are not used because $\pi_j \leq \pi_3$ for $j \geq 3$ by Lemma 2. The maximal energy consumption when using only O_2 and O_1 is calculated as follows:

$$e_2 = (R_2 - R_1) \cdot T_{3,2} + R_1 \cdot W$$

where $T_{3,2}$ denotes the value of π_2 such that $P_{\pi_2} = (R_2 - R_1)/(U_2 - U_1) \cdot (U_3 - U_2)/(R_3 - R_2)$. Similarly, the candidate solution using the set of $O_n, O_{n-1}, \ldots, O_1$ with the maximal energy consumption for $2 \leq n \leq N$ is found as follows:

$$P_{\pi_j} = 1 \cdot \frac{R_j - R_{j-1}}{U_j - U_{j-1}} \cdot \frac{U_n - U_{n-1}}{R_n - R_{n-1}}, \quad \text{for } 1 < j < n.$$

Its energy consumption is

$$e_N = \sum_{j=2}^{n} (R_j - R_{j-1}) \cdot T_{n+1,j} + R_1 \cdot W$$

where $T_{n+1,j}$ denotes the value of π_j such that $P_{\pi_j} = (R_j - R_{j-1})/(U_j - U_{j-1}) \cdot (U_n - U_{n-1})/(R_n - R_{n-1})$. Particularly, the candidate solution using only O_N for the worst time is found to measure the upper bound of achievable utility.

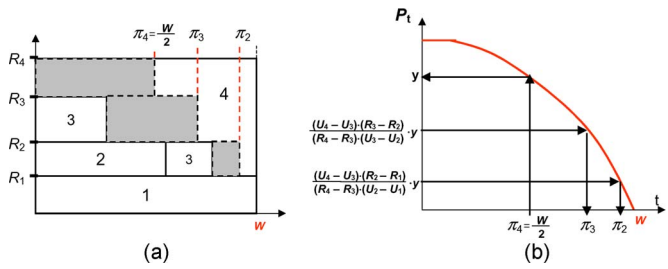

Fig. 4. Example of fine-grain partitioning procedure. (a) Creating an intermediate solution between CS_3 and CS_4. (b) Determining the transition times of O_4, O_3, and O_2.

The energy consumption when using only O_N for the time W is $e_{N+1} = R_N \cdot W$.

These $(N+1)$ candidate solutions are found for each task. The kth solution for the ith task is denoted as CS_k^i. The energy consumption of CS_k is smaller than that of CS_{k+1} for $1 \leq k \leq N$, as shown in Fig. 3. The energy consumption of CS_k is equivalent to the total areas of rectangles with an index no larger than k in Fig. 3. In CS_k such that $2 \leq k \leq N$, the starting and ending points of O_j are, respectively, $T_{k+1,j+1}$ and $T_{k+1,j}$ for $1 < j < k$. The procedure to find and store these $(N+1)$ candidate solution for each task is referred to as *Coarse-grain Partitioning*. The bisectional search operation to find the values of $T_{n+1,j}$s for $1 < j < n$ requires $O(N \cdot \log_2 W)$ steps for each candidate solution.

If $(e_{k+1} - e_k)/e_k > \epsilon$ for any k after completing the Coarse-grain Partitioning procedure, the online method creates more intermediate solutions between CS_k and CS_{k+1}. It creates an intermediate solution whose energy consumption is roughly equal to the median energy consumption of two existing adjacent solutions, CS_k and CS_{k+1}. As a base value for creating the intermediate solution, the middle point of any two transition time points π_hs found in CS_k and CS_{k+1} is fixed as the transition point π_h in the created intermediate solution. In the example of Fig. 4(a), the middle point $W/2$ of $\pi_4 = 0$ in CS_3 and $\pi_4 = W$ in CS_4 is fixed for the created intermediate solution. Based on the fixed π_h of the intermediate solution, it searches for the $(h-2)$ transition points of the other operation modes $O_{h-1}, \ldots, O_2$, i.e.,

$$P_{\pi_j} = \frac{R_j - R_{j-1}}{U_j - U_{j-1}} \cdot \frac{U_h - U_{h-1}}{R_h - R_{h-1}} \cdot P_{\pi_h}, \quad \text{for } 2 \leq j \leq (h-1)$$

such as shown in Fig. 4(b). The created intermediate solution is inserted between CS_k and CS_{k+1}. The procedure for finding an intermediate solution is called *Fine-grain Partitioning*. The complexity of the Fine-grain Partitioning procedure to search for at most N transition times in a bisectional manner is $O(N \cdot \log_2 W)$.

To prepare the final set of candidate solutions, the online method performs the Coarse-grain Partitioning procedure once and the Fine-grain Partitioning procedure several times[4] until $(e_{k+1}^i - e_k^i)/e_k^i \leq \epsilon$ for each k and i. For each candidate

[4]To find the Optimal Schedule discussed in Section III-A, the Fine-grain Partitioning procedure is repeated until the created intermediate solution consumes the given energy budget.

solution CS_k, the following values are calculated and stored with the values of e_k and π_ns to guide the runtime decisions.

ψ_k Output utility of CS_k.

ϕ_k Ratio of utility increment to additional energy from CS_{k-1} to CS_k, i.e., $\phi_k = (\psi_k - \psi_{k-1})/(e_k - e_{k-1})$.

The computation required to find candidate solutions is carried out once at the compilation time. Its computational complexity is $O(\sum_{i=1}^{M}\{\log_{(1+\epsilon)}(R^i_{N^i}/R^i_1)\cdot(N^i\cdot\log_2 W^i + W^i)\})$. For each task, the Coarse-grain Partitioning procedure is performed once and the Fine-grain Partitioning procedure is performed $O(\log_{(1+\epsilon)} R^i_{N^i}/R^i_1)$ times, where $(1+\epsilon)^\alpha \cdot R^i_1 \cdot W^i \simeq R^i_{N^i}\cdot W^i$ and $O(\log_{(1+\epsilon)} R^i_{N^i}/R^i_1)$ is the number of the generated candidate solutions. The computational complexities of the Coarse-grain Partitioning and the Fine-grain Partitioning procedures are $O(N^i\cdot\log_2 W^i)$. The operations to calculate e_k, ψ_k, and ϕ_k of each CS_k require $O(N^i)$, $O(W^i)$, and $O(1)$ steps, respectively.

The space complexity to store candidate solutions is $O(\sum_{i=1}^{M} N^i\cdot\log_{(1+\epsilon)} R^i_{N^i}/R^i_1)$ for all tasks, where $O(\log_{(1+\epsilon)} R^i_{N^i}/R^i_1)$ is the number of candidate solutions for each task and each candidate solution CS_k stores the values of e_k, ψ_k, ϕ_k and all π_ns. The space complexity to store the probability value for each discrete time is $O(\sum_{i=1}^{M} W^i)$. To reduce the space complexity, the same probability value for some period is managed with an interval element. Our method estimates the probabilistic execution time of a task from the records of its previous completion times. The previous completion times of each task are sorted in the ascending order and stored into an array or linked list. If the xth earliest completion time is c_x among the total A records of the task, the probabilistic execution time is calculated to be $P_t = (A-x-1)/A$ for $c_{x-1} < t \le c_x$. Then, the space complexity of storing the previous completion times into lists is $O(\sum_{i=1}^{M} A^i)$, where A^i is the number of recording the previous completion times for the ith task. Similarly, the computational complexity $O(\sum_{i=1}^{M}\{\log_{(1+\epsilon)}(R^i_{N^i}/R^i_1)\cdot(N^i\cdot\log_2 W^i + W^i)\})$ at the compilation time is replaced with $O(\sum_{i=1}^{M}\{\log_{(1+\epsilon)}(R^i_{N^i}/R^i_1)\cdot(N^i\cdot\log_2 A^i + N^i + A^i)\})$, because there are at most A^i different values for P^i_t and $(N^i + A^i)$ different values for $U^i_t\cdot P^i_t$. The number A^i is controllable.

B. Selection of Precomputed Solutions at Runtime

At runtime, the online method selects one candidate solution per task with the goal of maximizing the total utility of all selected solutions, subject to the constraint that the total energy consumption of all selected solutions is no larger than the given energy budget. To achieve this goal, the method proceeds as follows. It first selects the first solution CS^i_1 of each task. Then, $E^i = e^i_1$ for each i. Next, it searches for the solution CS^m_2 having the largest ratio of utility increment among the next solutions CS^i_2s behind the selected solutions CS^i_1s, i.e., $\phi^m_2 = \max_{1\le i\le M}\{\phi^i_2\}$. If substituting CS^m_2 for CS^m_1 does not exceed the given energy budget, CS^m_2 is selected for the mth task. Otherwise, it searches for the next largest ratio of utility increment until substituting the searched solution no longer exceeds the given energy budget. This process of substituting the next solution having the largest ratio of utility increment is repeated until any next solution exceeds the given energy budget.

The following pseudocode describes the runtime operation of the online method, called *Approximate Method*, where s^i and ΔE denote the index of the candidate solution selected for the ith task and the remaining available energy, respectively. Its computational complexity at runtime is $O(\sum_{i=1}^{M}\log_{(1+\epsilon)} R^i_{N^i}/R^i_1)$, where $O(\sum_{i=1}^{M}\log_{(1+\epsilon)} R^i_{N^i}/R^i_1)$ is the number of candidate solutions.

Approximate Method

Step 1) Select the first solution CS^i_1 of each task, i.e., $s^i \Leftarrow 1$ and $E^i \Leftarrow e^i_1$ for each i, and $\Delta E \Leftarrow E_{\max} - \sum_{i=1}^{M} E^i$.

Step 2) Search for the largest ratio $\phi^m_{s^m+1}$ among all $\phi^i_{s^i+1}$s. If $(e^m_{s^m+1} - e^m_{s^m}) \le \Delta E$, select the solution $CS^m_{s^m+1}$ for the mth task, i.e., $\Delta E \Leftarrow (\Delta E - (e^m_{s^m+1} - e^m_{s^m}))$ and $s^m \Leftarrow s^m + 1$.

Step 3) If $(e^m_{s^m+1} - e^m_{s^m}) > \Delta E$, search for the next largest $\phi^l_{s^l+1}$ until $(e^l_{s^l+1} - e^l_{s^l}) \le \Delta E$. Select the solution $CS^l_{s^l+1}$ for the lth task, i.e., $\Delta E \Leftarrow (\Delta E - (e^l_{s^l+1} - e^l_{s^l}))$ and $s^l \Leftarrow s^l + 1$.

Step 4) Repeat Steps 2) and 3) until $(e^i_{s^i+1} - e^i_{s^i}) > \Delta E$ for any i.

Step 5) Assign the selected solutions to the task scheduling.

If there are many candidate solutions, Step 2) of the Approximate Method may be repeated many times. To reduce the runtime iterations of Step 2), we can perform Step 2) at compile time until $\sum_{i=1}^{M} E^i = \sum_{i=1}^{M} R^i_{N^i}\cdot W^i$ and store the results of each iteration in a list. At runtime, the scheduler selects one result from the list, whose energy is the nearest to but no larger than $E_{\max}$, in a bisectional manner, and performs only Step 3) from the selected result.

To measure the utility difference between the Approximate Method and the Optimal Schedule associated with the energy $E_{\max}$, we define *Error Ratio* as

$$\frac{\Psi_{\text{opt}} - \Psi_{\text{app}}}{\Psi_{\text{opt}}}$$

where Ψ_{opt} and Ψ_{app} are the utility values of the Optimal Schedule and the Approximate Method, respectively. Theorem 2 in Appendix shows that the mean Error Ratio is smaller than the Error Bound ϵ for any $M \ge 1$. The value of Error Bound given by the user or the system administrator at compile time controls the tradeoff between the output quality and overhead of the Approximate Method. If the Error Bound is decreased, the Approximate Method provides a higher utility at the cost of larger computational overhead at compile time and larger memory overhead at runtime.

C. Rescheduling Invocations and Overhead of Changing Operation Modes

Whenever a task completes its execution much earlier than its worst execution time, the energy assigned to the task is partially used. The remainder energy can be exploited to improve the

utility of other tasks, if some tasks are still running or have not yet started at that point. The remainder energy is unknown in advance and dynamically given at the completion point. In the online method, the scheduler updates the operation modes of the uncompleted tasks with the increased energy at runtime. In addition, in order to minimize side-effects of errors in measuring the residual battery energy [14], [15], the scheduler periodically measures the residual energy and updates the instant operation modes. When $\Lambda_{\rm sch}$ and $D_{\rm sch}$ denote, respectively, the extra energy and time delay required to measure the residual available energy and perform rescheduling, the scheduler invokes the Approximate Method with $(E_{\max} - \Lambda_{\rm sch})$, to update the instant operation modes of tasks after time $D_{\rm sch}$. There is a tradeoff between the runtime overhead of rescheduling and the scheduling accuracy, which is beyond the scope of this paper.

Let us consider the overhead of changing the running operation mode. The extra energy and the time delay required to change the operation mode O_n with O_{n-1} are denoted as $\Lambda_{n,n-1}$ and $D_{n,n-1}$, respectively. In the candidate solution of the Approximate Method where $O_n, \ldots, O_1$ are used, its energy consumption e_k is replaced with $\hat{e_k} = (e_k + \sum_{j=2}^{n} \Lambda_{j,j-1})$. The operation modes selected with consideration of the energy overhead might differ from those selected without the energy overhead. When the nth candidate solution generated by the Coarse-grain Partitioning procedure uses $O_n, \ldots, O_1$ with energy consumption e_n, assigning the energy $\Lambda_{n+1,n}$ for the additional execution of $O_2, \ldots, O_n$ provides more utility than wasting the energy to replace O_{n+1} with O_n. If $(e_n + \sum_{j=2}^{n} \Lambda_{j,j-1}) < E^i$ (or $E_{\max}$) $\leq (e_n + \sum_{j=2}^{n+1} \Lambda_{j,j-1})$, the operation modes selected with the energy overhead are $O_1, \ldots, O_n$, whereas those without the energy overhead are $O_1, \ldots, O_{n+1}$. The values of $\pi_n, \ldots, \pi_2$ are increased until matching the replaced energy $\hat{e_n} = (e_n + \sum_{j=2}^{n+1} \Lambda_{j,j-1})$, instead of e_n.

The time delay $D_{n,n-1}$ affects the scheduling results, only when the utility values of operation modes correspond to the processing speed. In this case, the output utility ψ_k when using $O_n, \ldots, O_1$ is replaced with $\hat{\psi_k} = (\psi_k - \sum_{j=2}^{n} c_u \cdot D_{j,j-1})$ where c_u is a utility coefficient of delay. The utility $\hat{\psi_k}$ of a candidate solution using $O_n, \ldots, O_1$ might be smaller than the utility when using $O_{n-1}, \ldots, O_1$ with the same energy budget, due to the utility loss $c_u \cdot D_{n+1,n}$ wasted when changing O_{n+1} with O_n. Then, the selected operation modes $O_n, \ldots, O_1$ of the candidate solution are replaced with $O_{n-1}, \ldots, O_1$, and the running times of $O_{n-1}, \ldots, O_2$ increase.

V. Performance Evaluation

The Approximate Method is compared with the previous method [14], [15]. The previous method assigns the same operation mode to the whole processing parts regardless of their execution probability, while the Approximate Method assigns a higher utility mode to the processing parts being executed with a higher probability. For the evaluation metric, we define *Utility Increment* as

$$\frac{CU_o - CU_p}{CU_p} \times 100$$

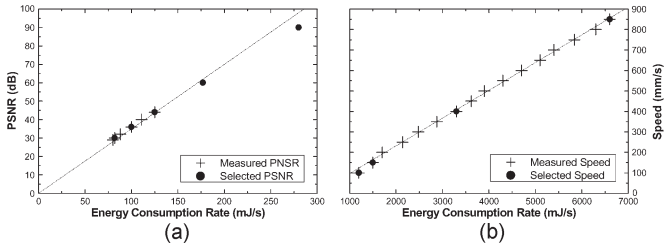

Fig. 5. Utility and energy consumption rate of measured data and selected data in (a) MPEG streaming and (b) Pioneer robot.

where CU_o and CU_p are the cumulative utility provided by our method and the previous method, respectively. This is the ratio of the additional utility created by our method to the output utility of the previous method. We also define *Feasible Energy* as

$$\frac{E_{\max} - E_l}{E_h - E_l}$$

where $E_l (= \sum_{m=1}^{M} R_1^m \cdot W^m)$ and $E_h (= \sum_{m=1}^{M} R_{N^m}^m \cdot W^m)$ are the minimum energy required to execute all tasks for the worst-case time in the lowest and highest operation mode, respectively. When the number of tasks or the energy difference between O_1 and O_N is large, the majority of full battery energy belongs to the range $[E_l, E_h]$ because the worst execution time is usually set to a sufficiently large value. Here, we do not consider the overhead required to change operation modes, because the previous study [14] showed that the overhead is negligible.

A. Models of Operation Modes and Varying Execution Times

To model a set of energy-aware operation modes, we draw the data from the operation modes of the MPEG-4 FGS streaming [2] and the Pioneer 3DX robot [9]. The video quality of the MPEG streaming in the optimized communication and the straight moving speed of the robot without load are used for their corresponding utility. Fig. 5 shows the relationship between their utility values and their energy consumption rates. Their utility values are almost linearly proportional to the energy consumption rates of the 802.11b WLAN card and the robot's motors as follows, respectively.

1) Energy rate of WLAN card (in millijoules per second) = $0.35 \times$ Peak Signal-to-Noise Ratio (PSNR) of MPEG streaming (in decibels).
2) Energy rate of robot's motors (in millijoules per second)= $7.4 \times$ Moving Speed (millimeters persecond) $+ 290$.

It is clear that the upper bound of Utility Increment is $(U_N - U_1)/U_1$. To avoid implicit side-effects of a low upper bound of Utility Increment, we select three data from the measured six data of the MPEG streaming and generate two data on the basis of the aforementioned linear increment, as shown in Fig. 5(a), while we simply select four data from the measured 16 data of the robot, as shown in Fig. 5(b).

To examine the impact of different distributions of task execution times, we use the data obtained from both real-life multimedia tasks and synthetically generated tasks. The execution times of multimedia tasks are drawn from the running time distribution of two sets of YouTube video clips (10K clips per set) [34] shown in Fig. 6(a), and the playing time distribution of

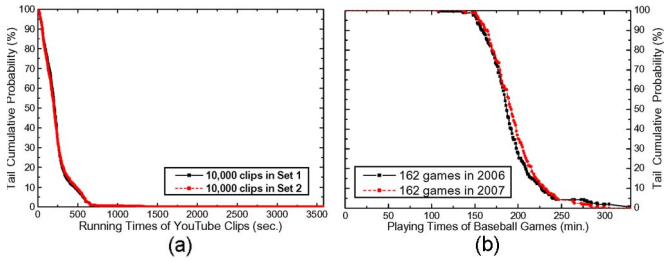

Fig. 6. Probability distribution of (a) the running times of YouTube clips and (b) the playing times of Major League Baseball games.

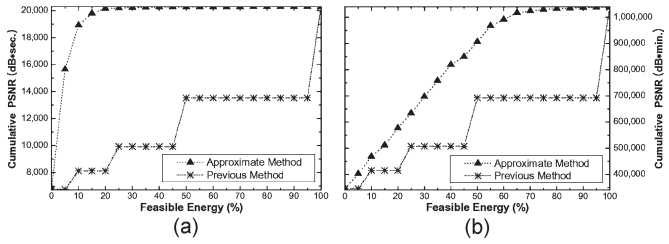

Fig. 7. Cumulative utility in (a) YouTube clips and (b) baseball games.

2006 and 2007 New York Yankees baseball games [35] shown in Fig. 6(b). The longest running time of the clips in set 1, 3580 s, is used for the worst time. In the case of baseball games, the longest playing time in 2006 is 330 min.

The execution times of synthetic tasks are generated between 1 and 1000 with normal, exponential, and uniform distributions. The worst execution time of all synthetic tasks is set to 1000. Those exceeding the worst time are truncated from the worst time. The mean value of the three distributions is initially given as 50% of the worst time. The standard deviation of the normal distribution is given as 10% of the mean value. In the performance evaluation, we run 1 000 000 synthetic tasks to obtain the steady mean utility of tasks having widely varying execution times, and display their averages.

B. Results of a Single Multimedia Task

Fig. 7 shows the total utility of the multimedia tasks, i.e., cumulative PSNR obtained for the whole MPEG-4 streaming session, with the five operation modes in Fig. 5(a). In the Approximate Method, the Error Bound ϵ is given as 0.05 (5%). Fig. 7(a) and (b) shows the averages of the cumulative PSNR in 10 000 runs for the clips of set 2, and in 162 runs for the 2007 baseball games, respectively. The cumulative utility of the Approximate Method in Fig. 7(a) increases more rapidly, compared with that in Fig. 7(b). This is mainly due to the relatively smaller mean of Fig. 6(a), where the Approximate Method executes earlier processing parts with a higher utility operation mode. In contrast, the increase patterns of the previous method in Fig. 7(a) and (b) are almost identical.

Fig. 8 shows Utility Increment of the Approximate Method and the difference from Utility Increment of the Optimal Schedule, denoted as "Error." The Approximate Method provides the largest Utility Increment 149% when 'Feasible Energy' $=$ 20% in Fig. 8(a), and 68% when "Feasible Energy" 'Feasible Energy' $=$ 45% in Fig. 8(b). Utility Increments in Fig. 8(a) are relatively larger than those in Fig. 8(b). When the mean of probabilistic execution time is smaller, the Approximate Method achieves better Utility Increment because it assigns a higher utility operation to the front processing parts and a lower utility operation to the rear processing parts than the previous method. In this figure, the Error values are smaller than 5% of Utility Increments in all cases. The Error value becomes almost zero when "Feasible Energy" $\geq$ 20% in Fig. 8(a) and when "Feasible Energy" $\geq$ 65% in Fig. 8(b). This is because the output utility approaches the upper bound of achievable utility when "Feasible Energy" $\geq$ 20% in Fig. 7(a), due to its relatively smaller mean execution time. The output utility approaches the upper bound when "Feasible Energy" $\geq$ 65% in Fig. 7(b), due to its relatively larger mean execution time.

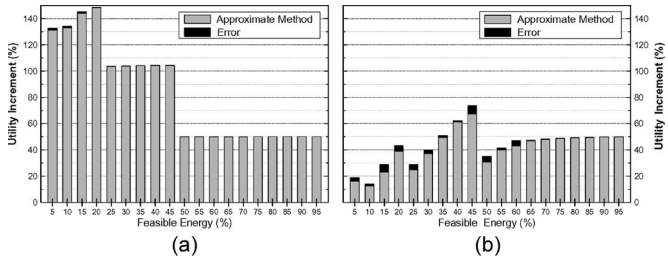

Fig. 8. Utility increments in (a) YouTube clips and (b) baseball games.

TABLE I
NUMBER OF CANDIDATE SOLUTIONS

Error Bound	3%	5%	10%	20%	30%
YouTube clips	95	69	45	28	23
Baseball games	66	41	22	14	12

Table I shows the number of candidate solutions prepared by the Approximate Method at compile time, when $\epsilon = 3\%$, $\epsilon = 5\%$, $\epsilon = 10\%$, $\epsilon = 20\%$, and $\epsilon = 30\%$. As Error Bound decreases, the number of candidate solutions increases. The number of solutions for YouTube clips is relatively larger than that for baseball games. The number of solutions using all five operation modes for YouTube clips is much larger than that for baseball games, while the number of solutions using four or fewer operation modes for YouTube clips is equal to or slightly smaller than that for baseball games. This is because the energy difference between the two adjacent solutions CS_5 and CS_6 generated by the Coarse-grain Partitioning procedure for YouTube clips is larger than that for baseball games. The energy consumption of CS_5 for YouTube clips is about 7% of Feasible Energy, while that for baseball games is about 24% of Feasible Energy. The energy consumption of CS_6 is 100% of Feasible Energy in both cases.

To explore the effect of different utility settings, we vary the number of available operation modes. Fig. 9(a) and (b) shows the results of the YouTube clips when using six operation modes with an additional operation mode $O_6 = [R_6 = 502, U_6 = 150]$ and when using four operation modes without O_5, respectively. Comparing these results with those of Fig. 8(a) confirms that the proposed method achieves better Utility Increment when the largest utility difference between available operation modes, i.e., U_N/U_1, increases. This is because the Approximate Method assigns the highest utility mode to the front processing parts and the lowest utility mode to the rear processing parts. The best values of Utility Increment in

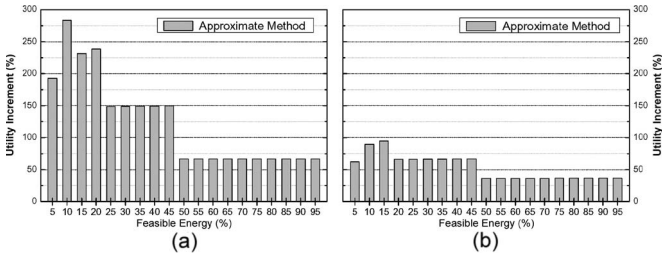

Fig. 9. Utility increment for (a) six operation modes with additional $O_6 = [R_6 = 502, U_6 = 150]$ and (b) four operation modes without O_5.

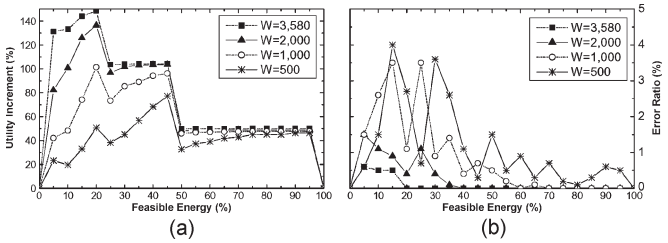

Fig. 10. (a) Utility increment and (b) error ratio of the approximate method with respect to the worst-case value.

Figs. 8(a), 9(a) and (b) are 149%, 284%, and 95%, respectively, which are approximately 50% of U_N/U_1.

To examine the impact of different worst-case settings, we vary the worst time of the YouTube clips. Fig. 10(a) shows Utility Increment when the worst execution time is 500, 1000, 2000, or 3580. The execution times exceeding the worst time are truncated from the worst time. As the worst time decreases, the mean execution time of the YouTube clips approaches the worst time. The Approximate Method achieves better utility enhancement when the mean execution time is smaller, for the same reason shown in Fig. 8. Fig. 10(b) shows Error Ratio of the Approximate Method when Error Bound $= 5\%$. Error Ratio decreases as the worst time increases, because Utility Increment of the longer worst time is larger, as shown in Fig. 10(a), while the difference to the maximum utility is almost the same regardless of the worst time. The values of Error Ratio on low Feasible Energy are relatively larger than those on high Feasible Energy. This is because the probability P_t increases rapidly on low Feasible Energy but rarely on high Feasible Energy, as shown in Fig. 7(a). The larger worst time expedites the profitable capability to support the completion of a very long execution case exceeding the estimated maximum time. Note that, if the worst execution time is set to a small value, the risk of terminating an unexpectedly long execution before completion due to energy depletion is increased. Therefore, we expect the worst execution time to be sufficiently large, in which case our scheme is more likely to attain better performance.

C. Results of Two Synthetic Tasks

In this section, we examine the performance when two tasks are running concurrently. The first task is executed over the five operation modes of Fig. 5(a) and the second task is executed over the four operation modes of Fig. 5(b). The tasks are generated with a normal, exponential, or uniform distribution. When one task completes its execution earlier than the other task, the remaining energy of the completed task is utilized for higher

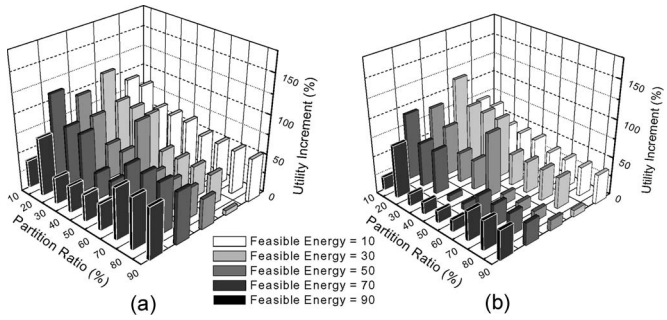

Fig. 11. Utility increments versus partition ratio of feasible energy when generating a task with (a) normal distribution and (b) exponential distribution.

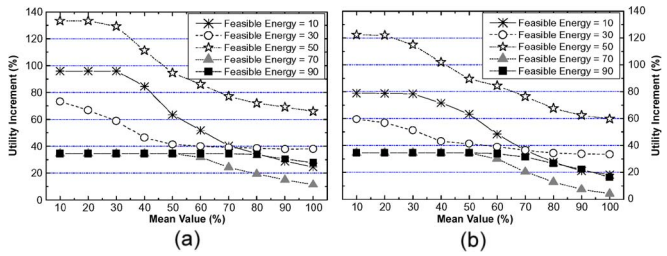

Fig. 12. Utility increments with respect to various means when generating tasks with (a) accurate and (b) inaccurate probability estimation.

utility operation of the other task from the completion point. Because the previous method [14], [15] has no mechanism for distributing available energy to multiple tasks, we manually distribute the available energy to the two tasks in implementation of the previous method. Recall that the Approximate Method systematically determines the distribution ratio of the available energy. The output utility of the Approximate Method is identical for a given amount of available energy, while that of the previous method changes according to the decision about how to distribute the available energy to the two tasks.

Fig. 11 shows the total Utility Increment of the two tasks, where "Partition Ratio" denotes the ratio of the energy amount used by the first task to the total energy in the previous method. The first task is generated with a normal distribution in Fig. 11(a) and an exponential distribution in Fig. 11(b). The second task is generated with a uniform distribution in both Fig. 11(a) and (b). Their mean execution times are the half of their worst execution times. The execution times used to calculate P_t are generated with the same distribution for each task. It is interesting that the best case of Partition Ratio in the previous method changes according to the Feasible Energy. This is because available operation modes are discrete and have different ratios of their utility values to their energy consumptions. In this experiment, the Error Bound of the Approximate Method is given as 5% and the Error Ratio is smaller than 5% in all cases.

Fig. 12 shows the results when the two tasks have various mean execution times, denoted as the ratio of the worst execution time. In this experiment, the first task is generated with a normal distribution and the second task is generated with an exponential distribution. In the previous method, the available energy is evenly distributed to the two tasks. In Fig. 12(a), the execution times of past tasks used to calculate P_t have the same mean as those of running tasks. The Approximate Method achieves a higher Utility Increment when the mean execution

time is smaller, because it assigns a higher utility operation to the front processing parts and a lower utility operation to the rear processing parts than the previous method. When Feasible Energy $= 70\%$ or 90%, the benefit of the Approximate Method is diminished because the previous method also assigns a higher utility operation if there is more available energy.

To examine the robustness of the Approximate Method with respect to inaccurate estimation of the probabilistic execution time, we fix the mean execution time of past tasks to 50% of the worst time and vary the mean execution time of running tasks in Fig. 12(b). Comparing the results of Fig. 12(a) with those of Fig. 12(b), the difference of their Utility Increments increases as the difference of their means increases. The Approximate Method achieves a larger utility when it is constructed with a more exact probabilistic execution time. When the mean execution time of running tasks is smaller than 50% and Feasible Energy is equal to 70% or 90%, Utility Increments in Fig. 12(a) and (b) are almost the same. If there is large amount of available energy, the front processing parts are rarely sensitive to the accuracy of probabilistic execution time, because the Approximate Method always assigns the highest utility operation mode to the front parts when there is large amount of energy.

VI. Related Work

Some studies [14]–[17] have considered the performance degradation as a means to maintain the battery operation for a specified critical duration. Graceful degradation [14], [15], [17] or stopping unimportant tasks [16] when the available energy is low can allow critical tasks to run for the specified duration. Lee *et al.* [36], [37] addressed the problem of maximizing the output utilities of multiple tasks executed under resource constraints, where a utility consists of multidimensional values. They showed that this problem is NP-hard, and proposed polynomial-time iterative algorithms to find a suboptimal solution. Recent studies [38], [39] addressed the problem of maximizing the output utility of a real-time task while satisfying energy and real-time constraints. Unfortunately, however, all of these studies only considered the energy management of tasks with their fixed execution time, i.e., the worst execution time, without considering a wide range of task execution times. Moreover, the proportion of energy distributed to multiple tasks is determined manually, due to the absence of a systematic distribution mechanism of available energy.

More recent studies [40], [41] addressed a similar approach that maximizes the output utility of extra computation obtained by dynamically exploiting the slack energy resulting from the variation of execution times. This method uses the slack energy of preceding tasks, completed earlier than their schedules, for additional computation of succeeding tasks over infinitely continuous operation modes. Our approach contrasts with the work in two aspects: it can work during the execution of *concurrent* tasks, and it considers *discrete* operation modes more realistically. The study for *imprecise computation* model [42] differs from this paper, in that it dynamically exploits reclaimed resources, left after simply completing all *mandatory* tasks, for *optional* tasks, whereas our method stochastically exploits given resources only for *mandatory* tasks.

Some studies of real-time scheduling [10], [11], [18]–[21] have utilized a probability distribution of task execution times for the minimal energy consumption of CPU processing in the DVFS framework. These methods might be applicable to the problem tackled in this paper with some modifications: replacing the deadline constraint with the energy constraint, and replacing the energy minimization of CPU with the utility maximization of a task. However, these methods suffer from four drawbacks: infinitely continuous operation modes [10], [11], [18], [21], an enforced relationship between the execution speed and energy consumption of operation modes [10], [11], [18], [21], consideration of only a single task [10], [11], [18]–[20], and heavy computational overhead [19], [20]. Contrarily, our online method yields a near optimal and adjustable solution for *multiple* tasks with *little runtime* overhead over *finitely discrete* operation modes providing *different utilities per unit energy consumption*.

Finally, our previous work [32] maximizes the cumulative mean utility of a single task over continuous operation modes with an enforced relationship between utility U and energy consumption rate R, i.e., $U = (R)^{\alpha}$ for any $0 < R$ and $0 < \alpha$.

VII. Conclusion

For mobile wireless systems relying on limited energy budgets, we propose an optimal offline scheduling scheme which stochastically maximizes the cumulative utility of multiple tasks with probabilistic execution times. In order to diminish the heavy computational overhead of the optimal offline scheme, we also propose an approximate online scheme which operates with little runtime overhead and provides almost the maximum utility. Extensive evaluation shows that the utility values of the optimal offline scheme and the approximate online scheme are very similar in most cases, and the online scheme provides much higher utility than the previous methods. The proposed schemes can be applied to other problems that need to maximize the cumulative utility of multiple tasks with probabilistic execution times subject to a single resource constraint. An example problem is maximization of the cumulative communication quality of a voice call with a limited prepayment deposit, where the charge per unit time depends on the communication quality and voice calls have a wide range of communication times. In future work, we will investigate the scheduling scheme that changes the operation modes of dependent devices affected by the operation modes of other devices.

Appendix

Lemma 1: The Optimal Schedule excludes the operation mode O_y such that $(U_y - U_x)/(R_y - R_x) < (U_z - U_y)/(R_z - R_y)$ for any $R_x < R_y < R_z$ (and $U_x < U_y < U_z$).

Proof: Let us assume that the Optimal Schedule uses the operation mode O_y from time t_1 to time t_2 for any $0 \le t_1 < t_2 \le W$. When $\tau = (t_1 + ((R_y - R_x) \cdot (t_2 - t_1))/(R_z - R_x))$ such that $t_1 < \tau < t_2$, we can generate another schedule by replacing O_y with O_z from time t_1 to time τ and replacing O_y with O_x from time τ to time t_2. Then

$$(t_2 - t_1) \cdot R_y = (\tau - t_1) \cdot R_z + (t_2 - \tau) \cdot R_x$$

which means that the generated schedule consumes the same amount of energy with the Optimal Schedule. If $(U_y - U_x)/(R_y - R_x) < (U_z - U_y)/(R_z - R_y)$, then

$$(\tau - t_1) \cdot U_y + (t_2 - \tau) \cdot U_y < (\tau - t_1) \cdot U_z + (t_2 - \tau) \cdot U_x$$

and thus

$$\int_{t_1}^{\tau} U_y \cdot P_t dt + \int_{\tau}^{t_2} U_y \cdot P_t dt < \int_{t_1}^{\tau} U_z \cdot P_t dt + \int_{\tau}^{t_2} U_x \cdot P_t dt$$

where $P_{t_1} \geq P_{t_2}$ for $t_1 < t_2$, i.e., the generated schedule provides more utility than the Optimal Schedule. This is a contradiction on the definition of the Optimal Schedule. Hence, the Optimal Schedule excludes the operation mode O_y. ■

Lemma 2: In an Optimal Schedule, $\mathcal{F}_u(\mathcal{SO}_{t_1}) \geq \mathcal{F}_u(\mathcal{SO}_{t_2})$ and $\mathcal{F}_r(\mathcal{SO}_{t_1}) \geq \mathcal{F}_r(\mathcal{SO}_{t_2})$ for any $0 < t_1 < t_2 \leq W$.

Proof: When $O_x = \mathcal{SO}_{t_1}$ and $O_y = \mathcal{SO}_{t_2}$ for any $0 < t_1 < t_2 \leq W$, let us assume that $R_x < R_y$ and $U_x < U_y$ in the Optimal Schedule. We can generate another schedule to assign O_x to the time t_2 and O_y to the time t_1. By the definition of P_t in Section II-B, $P_{t_1} \geq P_{t_2}$ for $t_1 < t_2$. If $P_{t_1} > P_{t_2}$, the generated schedule provides more utility than the assumed Optimal Schedule with the same energy consumption. This is a contradiction on the definition of the Optimal Schedule. If $P_{t_1} = P_{t_2}$, the generated schedule is another Optimal Schedule because it provides the same utility and consumes the same energy with the assumed Optimal Schedule. Hence, $\mathcal{F}_u(\mathcal{SO}_{t_1}) \geq \mathcal{F}_u(\mathcal{SO}_{t_2})$ and $\mathcal{F}_r(\mathcal{SO}_{t_1}) \geq \mathcal{F}_r(\mathcal{SO}_{t_2})$ for any $0 < t_1 < t_2 \leq W$ in an Optimal Schedule. ■

Theorem 1: The values of π_ns in an Optimal Schedule obey the following relationship:

$$P_{\pi_x} \cdot \frac{U_x - U_{x-1}}{R_x - R_{x-1}} = P_{\pi_y} \cdot \frac{U_y - U_{y-1}}{R_y - R_{y-1}}$$

for $2 \leq x < y \leq N$ such that $W \geq \pi_x > \pi_y > 0$.

Proof: Let us apply the Lagrange Multiplier Method [33] to (3) in Section III-A. When

$$\begin{aligned} L(\pi_N, \ldots, \pi_2, \lambda) &= \sum_{n=2}^{N} (U_n - U_{n-1}) \cdot \int_0^{\pi_n} P_t dt + U_1 \cdot \int_0^{W} P_t dt + \lambda \\ &\quad \cdot \left\{ E_{\max} - \left(\sum_{n=2}^{N} (R_n - R_{n-1}) \cdot \pi_n + R_1 \cdot W \right) \right\} \\ \frac{\partial L}{\partial \lambda} &= E_{\max} - \left(\sum_{n=2}^{N} (R_n - R_{n-1}) \cdot \pi_n + R_1 \cdot W \right) \\ &= 0 \\ \frac{\partial L}{\partial \pi_n} &= \frac{(U_n - U_{n-1}) \cdot \int_0^{\pi_n} P_t dt}{\partial \pi_n} - \lambda \cdot (R_n - R_{n-1}) \\ &= 0 \quad \text{for } 2 \leq n \leq N. \end{aligned}$$

Then

$$\frac{\int_0^{\pi_n} P_t \, dt}{\partial \pi_N} = P_{\pi_N} = \lambda \cdot \frac{R_n - R_{n-1}}{U_k - U_{n-1}}, \quad \text{for } 2 \leq n \leq N$$

and thus the value of P_{π_n} is strictly proportional to the value of $(R_n - R_{n-1})/(U_n - U_{n-1})$. Hence, $P_{\pi_x} \cdot (U_x - U_{x-1})/(R_x - R_{x-1}) = P_{\pi_y} \cdot (U_y - U_{y-1})/(R_y - R_{y-1})$ for any $2 \leq x < y \leq N$. ■

Lemma 3: The utility of the Optimal Schedule constructs a concavely increasing function with the input of $E_{\max}$.

Proof: The transition time π_n of O_n increases with increasing $E_{\max}$, because $P_{\pi_n} \cdot (U_n - U_{n-1})/(R_n - R_{n-1}) = P_{\pi_{n-1}} \cdot (U_{n-1} - U_{n-2})/(R_{n-1} - R_{n-2})$ for each n by Theorem 1 and $E_{\max} = (\sum_{n=2}^{N} (R_n - R_{n-1}) \cdot \pi_n + R_1 \cdot W)$. Let π_n^a and π_n^b denote the transition times of O_n in the Optimal Schedule when using the energy of $E_{\max}$ and the energy of $(E_{\max} + \Delta E)$ for arbitrary $\Delta E > 0$, respectively. The utility increment gained with the additional energy ΔE in the Optimal Schedule is

$$\sum_{n=2}^{N} (U_N - U_{n-1}) \cdot \int_{\pi_N^a}^{\pi_N^b} P_t dt \tag{5}$$

where $\Delta E = \sum_{n=2}^{N} (R_n - R_{n-1}) \cdot (\pi_n^b - \pi_n^a)$. Because $P_{t_1} \geq P_{t_2}$ for $t_1 < t_2$ and the value of π_n increases with increasing $E_{\max}$ for each n, the value ratio of $\int_{\pi_n^a}^{\pi_n^b} P_t dt$ to $(\pi_n^b - \pi_n^a)$ decreases with increasing $E_{\max}$ for each n. Accordingly, the value ratio of (5) to ΔE decreases with increasing $E_{\max}$. This means that the utility of the Optimal Schedule constructs a concavely increasing function with the input of $E_{\max}$. ■

Lemma 4: When $M = 1$, the mean Error Ratio is smaller than the Error Bound.

Proof: When $e_k \leq E_{\max} < e_{k+1}$, the Approximate Method provides the utility of $\Psi_{\text{app}} = \psi_k$ and the Optimal Schedule provides the utility of Ψ_{opt} such that $\psi_k \leq \Psi_{\text{opt}} < \psi_{k+1}$. Then

$$\frac{\Psi_{\text{opt}} - \Psi_{\text{app}}}{\Psi_{\text{opt}}} < \frac{\psi_{k+1} - \psi_k}{\Psi_{\text{opt}}} < \frac{\psi_{k+1} - \psi_k}{\psi_k}.$$

Because the value of ψ_k constructs a concavely increasing function with the input of e_k by Lemma 3, $\psi_k/e_k \geq \psi_{k+1}/e_{k+1}$ and thus $\psi_k/\psi_{k+1} \geq e_k/e_{k+1}$. Then

$$\frac{\psi_{k+1} - \psi_k}{\psi_k} \leq \frac{e_{k+1} - e_k}{e_k} \leq \epsilon. \tag{6}$$

Hence, the mean Error Ratio $(\Psi_{\text{opt}} - \Psi_{\text{app}})/\Psi_{\text{opt}}$ is smaller than the Error Bound ϵ. ■

Theorem 2: The mean Error Ratio is smaller than the Error Bound for any $M \geq 1$.

Proof: Lemma 4 shows that the mean Error Ratio of the Approximate Method is smaller than ϵ when $M = 1$. Consider the case when $M \geq 2$. We use the following notations for simplicity.

E_{opt}^i and E_{app}^i Values of E^i assigned to the ith task by the Optimal Schedule and the Approximate Method, respectively.

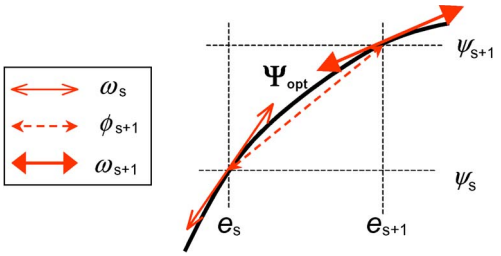

Fig. 13. Relationship among the values of $\omega^i_{s^i}$, $\phi^i_{s^i+1}$, and $\omega^i_{s^i+1}$.

$\Psi^i_{\rm opt}$ and $\Psi^i_{\rm app}$ Utility values gained from the ith task in the Optimal Schedule and the Approximate Method, respectively.

$\omega^i_{\rm opt}$ Instant derivative of $\Psi^i_{\rm opt}$ with regard to the value of $E^i_{\rm opt}$.

$\omega^i_{s^i}$ Instant derivative of $\Psi^i_{\rm app}$ with regard to the value of $E^i_{\rm app}$, when the Approximate Method finally selects the sth solution precomputed for the ith task.

Then, $E_{\max} = \sum_{i=1}^{M} E^i_{\rm opt}$, $E_{\max} \geq \sum_{i=1}^{M} E^i_{\rm app}$, $\Psi_{\rm opt} = \sum_{i=1}^{M} \Psi^i_{\rm opt}$, and $\Psi = \sum_{i=1}^{M} \Psi^i_{\rm app}$. In addition, $E^i_{\rm opt} \geq E^i_{\rm app}$ for each i, because the Optimal Schedule assigns additional energy to the task having the highest derivative until $E_{\max} = \sum_{i=1}^{M} E^i_{\rm opt}$.

Lemma 3 shows that $\omega^i_{\rm opt}$ decreases as $E^i_{\rm opt}$ increases for each i. Then, $\omega^i_{s^i} \geq \omega^i_{\rm opt} \geq \omega^i_{s^i+1}$ for each i, where s^i denotes the index of the precomputed solution finally selected by the Approximate Method for the ith task. When ϕ^i_k denotes the utility increment ratio of the kth solution precomputed for the ith task against the energy increment (as described in Section IV-A)

$$\omega^i_{s^i} \geq \phi^i_{s^i+1} \geq \omega^i_{s^i+1}, \quad \text{for each } i$$

such as shown in Fig. 13. Because the Approximate Method preferentially selects the precomputed solution having the largest value of ϕ^i

$$\omega^i_{s^i} \geq \max_{1\leq j\leq M} \left\{\phi^j_{s^j+1}\right\} \geq \max_{1\leq j\leq M} \left\{\omega^j_{s^j+1}\right\}, \quad \text{for each } i.$$

When ψ^i_k denotes the output utility of the kth solution precomputed for the ith task (as described in Section IV-A)

$$\Psi^i_{\rm opt} < \psi^i_{s^i+1} \text{ and } (\Psi_{\rm opt} - \Psi_{\rm app}) < \sum_{i=1}^{M} \left(\psi^i_{s^i+1} - \psi^i_{s^i}\right).$$

Then

$$\frac{\Psi_{\rm opt} - \Psi_{\rm app}}{\Psi_{\rm opt}} \leq \frac{\Psi_{\rm opt} - \Psi_{\rm app}}{\Psi_{\rm app}} < \frac{\sum_{i=1}^{M} \left(\Psi^i_{\rm opt} - \Psi^i_{\rm app}\right)}{\sum_{i=1}^{M} \Psi^i_{\rm app}}.$$

Because $(\psi^i_{s^i+1} - \psi^i_{s^i})/\psi^i_{s^i} \leq \epsilon$ by (6) in Lemma 4

$$\frac{\sum_{i=1}^{M} \left(\Psi^i_{\rm opt} - \Psi^i_{\rm app}\right)}{\sum_{i=1}^{M} \Psi^i_{\rm app}} \leq \epsilon.$$

Consequently, $(\Psi_{\rm opt} - \Psi_{\rm app})/\Psi_{\rm opt} < \epsilon$ for any $M \geq 1$. ■

REFERENCES

[1] C. Poellabauer and K. Schwan, "Energy-aware media transcoding in wireless systems," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, Mar. 2004, pp. 135–144.

[2] K. Choi, K. Kim, and M. Pedram, "Energy-aware MPEG-4 FGS streaming," in *Proc. DAC*, 2003, pp. 912–915.

[3] Q. Qin, Q. Wu, and M. Pedram, "Dynamic power management in a mobile multimedia system with guaranteed quality-of-service," in *Proc. DAC*, Jun. 2001, pp. 834–839.

[4] G. Anastasi, A. Passarella, M. Conti, E. Gregori, and L. Pelusi, "A power-aware multimedia streaming protocol for mobile users," in *Proc. IEEE Int. Conf. Pervasive Serv.*, Jul. 2005, pp. 371–380.

[5] Y.-H. Lu, L. Benini, and G. De Micheli, "Power-aware operating systems for interactive systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 2, pp. 119–134, Apr. 2002.

[6] L. D. Paulson, "TV comes to the mobile phone," *Computer*, vol. 39, no. 4, pp. 13–16, Apr. 2006.

[7] R. Kravets and P. Krishnan, "Power management techniques for mobile communication," in *Proc. Annu. ACM/IEEE Int. Conf. Mobile Comput. Netw. (MobiCom)*, 1998, pp. 157–168.

[8] J. Brateman, C. Xian, and Y.-H. Lu, "Energy-efficient scheduling for autonomous mobile robots," in *Proc. IFIP Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, Oct. 2006, pp. 361–366.

[9] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "A case study of mobile robot's energy consumption and conservation techniques," in *Proc. Int. Conf. Adv. Robot.*, Jul. 2005, pp. 492–497.

[10] J. R. Lorch and A. J. Smith, "PACE: A new approach to dynamic voltage scaling," *IEEE Trans. Comput.*, vol. 53, no. 7, pp. 856–869, Jul. 2004.

[11] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time CPU scheduling for mobile multimedia systems," in *Proc. ACM Symp. Oper. Syst. Principles*, Oct. 2003, pp. 149–163.

[12] A. Davids, "Urban search and rescue robots: From tragedy to technology," *IEEE Intell. Syst.*, vol. 17, no. 2, pp. 81–83, Mar./Apr. 2002.

[13] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 301–317, Mar. 2001.

[14] J. Flinn and M. Satyanarayanan, "Managing battery lifetime with energy-aware adaptation," *ACM Trans. Comput. Syst.*, vol. 22, no. 2, pp. 137–179, May 2004.

[15] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat, "ECOSystem: Managing energy as a first class operating systems resource," *ACM SIGOPS Oper. Syst. Rev.*, vol. 30, no. 5, pp. 123–132, Dec. 2002.

[16] M. Tamai, T. Sun, K. Yasumoto, N. Shibata, and M. Ito, "Energy-aware video streaming with QoS control for portable computing devices," in *Proc. ACM Int. Workshop Netw. Oper. Syst. Support Digital Audio Videos*, 2004, pp. 68–73.

[17] W. Yuan, K. Nahrstedt, S. V. Adve, D. L. Jones, and R. H. Kravets, "GRACE-1: Cross-layer adaptation for multimedia quality and battery energy," *IEEE Trans. Mobile Comput.*, vol. 5, no. 7, pp. 799–815, Jul. 2006.

[18] J. A. Barnett, "Dynamic task-level voltage scheduling optimizations," *IEEE Trans. Comput.*, vol. 54, no. 5, pp. 508–520, May 2005.

[19] R. Xu, C. Xi, R. Melhem, and D. Mossé, "Practical PACE for embedded systems," in *Proc. ACM Int. Conf. EMSOFT*, Sep. 2004, pp. 54–63.

[20] Z. Lu, Y. Zhang, M. Stan, J. Lach, and K. Skadron, "Procrastinating voltage scheduling with discrete frequency sets," in *Proc. DATE Conf.*, 2006, pp. 456–461.

[21] C. Xian, Y.-H. Lu, and Z. Li, "Dynamic voltage scaling for multitasking real-time systems with uncertain execution time," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 8, pp. 1467–1478, Aug. 2008.

[22] L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 1548–1557.

[23] Z. Zhou, P. K. McKinley, and S. M. Sadjadi, "On quality-of-service and energy consumption tradeoffs in FEC-enabled audio streaming," in *Proc. IEEE IWQoS*, Montreal, QC, Canada, 2004, pp. 161–170.

[24] H. Shim, N. Chang, and M. Pedram, "A backlight power management framework for battery-operated multimedia systems," *IEEE Des. Test Comput.*, vol. 21, no. 5, pp. 388–396, Sep./Oct. 2004.

[25] A. R. Lebeck, X. Fan, H. Zeng, and C. Ellis, "Power aware page allocation," *ACM SIGOPS Oper. Syst. Rev.*, vol. 34, no. 5, pp. 105–116, Dec. 2000.

[26] F. Douglis, P. Krishnan, and B. N. Bershad, "Adaptive disk spin-down policies for mobile computers," in *Proc. Symp. MLICS*, 1995, pp. 121–137.

[27] S. Mohapatra, N. Dutt, A. Nicolau, and N. Venkatasubramanian, "DYNAMO: A cross-layer framework for end-to-end QoS and energy optimization in mobile handheld devices," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 4, pp. 722–737, May 2007.
[28] C. Krintz, Y. Wen, and R. Wolski, "Application-level prediction of battery dissipation," in *Proc. ISLPED*, 2004, pp. 224–229.
[29] D. Rakhmatov, S. Vrudhula, and D. A. Wallach, "Battery lifetime prediction for energy-aware computing," in *Proc. ISLPED*, 2002, pp. 154–159.
[30] Y. Wen, R. Wolski, and C. Krintz, "Online prediction of battery lifetime for embedded and mobile devices," in *Proc. Int. Workshop PACS*, Dec. 2003, vol. 3164, pp. 57–72.
[31] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J.-S. Liu, "Probabilistic performance guarantee for real-time tasks with varying computation times," in *Proc. IEEE RTAS*, May 1995, pp. 164–173.
[32] W. Y. Lee, H. Lee, and H. Kim, "Energy-aware QoS adjustment of multimedia tasks with uncertain execution time," in *Proc. ICCS*, May 2007, vol. 4490, pp. 709–712.
[33] D. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison-Wesley, 1984.
[34] *Featured videos*. [Online]. Available: http://www.youtube.com
[35] ESPN, *MLB scoreboard*. [Online]. Available: http://sports-ak.espn.go.com/mlb/scoreboard
[36] C. Lee, J. Lehoczky, R. Rajkumar, and D. Siewiorek, "On quality of service optimization with discrete QoS options," in *Proc. IEEE RTAS*, Jun. 1999, pp. 276–286.
[37] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen, "A scalable solution to the multi-resource QoS problem," in *Proc. IEEE RTSS*, 1999, pp. 315–326.
[38] C. Rusu, R. Melhem, and D. Mossé, "Maximizing rewards for real-time applications with energy constraints," *ACM Trans. Embed. Comput. Syst.*, vol. 2, no. 4, pp. 537–559, Dec. 2003.
[39] P. Pillai, H. Huang, and K. G. Shin, "Energy-aware quality of service adaptation," Univ. Michigan, Ann Arbor, MI, Tech. Rep. CSE-TR-479-03, 2003.
[40] C. Rusu, R. Melhem, and D. Mossé, "Multi-version scheduling in rechargeable energy-aware real-time systems," *J. Embed. Comput.*, vol. 1, no. 2, pp. 271–283, Apr. 2005.
[41] L. A. Cortés, P. Eles, and Z. Peng, "Quasi-static assignment of voltages and optional cycles in imprecise-computation systems with energy considerations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 10, pp. 1117–1129, Oct. 2006.
[42] J. W. S. Liu, K.-J. Lin, W.-K. Shih, A. C.-S. Yu, J.-Y. Chung, and W. Zhao, "Algorithms for scheduling imprecise computations," *Computer*, vol. 24, no. 5, pp. 58–68, May 1991.

Wan Yeon Lee received the B.S., M.S., and Ph.D. degrees in computer science and engineering from Pohang University of Science and Technology, Gyungbuk, Korea, in 1994, 1996, and 2000, respectively.

From 2000 to 2003, he was a Research Engineer with LG Electronics and worked for the standardization of Next Generation Mobile Network of 3GPP Group. From 2006 to 2007, he was a Visiting Professor with the School of Electrical Engineering, Purdue University, West Lafayette, IN. He is currently an Associate Professor with the Department of Computer Engineering, Hallym University, Chunchon, Korea. His research interest includes embedded system, real-time system, mobile computing, computer security, and parallel computing.

Hyogon Kim received the Ph.D. degree from the University of Pennsylvania, Philadelphia, in 1995.

He is currently a Professor with the Korea University, Seoul, Korea. Prior to joining the Korea University, he was a Research Scientist with Bell Communications Research (Bellcore). His research interests include Internet Protocols and applications, wireless networking, and network security.

Heejo Lee (M'04) received the B.S., M.S., and Ph.D. degrees in computer science and engineering from Pohang University of Science and Technology, Gyungbuk, Korea.

From 2000 to 2001, he was a Postdoctorate with the Department of Computer Sciences and the Center for Education and Research in Information Assurance and Security, Purdue University, West Lafayette, IN. From 2001 to 2003, he was a Chief Technical Officer with AhnLab, Inc. He is currently an Associate Professor with the Division of Computer and Communication Engineering, Korea University, Seoul, Korea.

Dr. Lee serves as an Editor of the *Journal of Communications and Networks*. He has been an advisory member of Korea Information Security Agency and Korea Supreme Prosecutor's Office.